UNITED STATES PATENT APPLICATION

of

Yakov Rekhter

and

Eric C. Rosen

for

PEER-MODEL SUPPORT FOR VIRTUAL PRIVATE NETWORKS HAVING

POTENTIALLY OVERLAPPING ADDRESSES

PEER-MODEL SUPPORT FOR VIRTUAL PRIVATE NETWORKS HAVING POTENTIALLY OVERLAPPING ADDRESSES

CROSS-REFERENCE TO RELATED APPLICATIONS

5   The present application is a continuation of commonly assigned copending U.S. Patent Application Serial No. 08/997,343, which was filed on December 23, 1997, by Yakov Rekhter and Eric C. Rosen for Peer-Model Support for Virtual Private Networks with Potentially Overlapping Addresses.

BACKGROUND OF THE INVENTION

10   *Field of the Invention*

The present invention is directed to communications networking. It is directed particularly to providing routing for private wide-area networks.

Background Information

An enterprise that has many sites can build a private wide-area network by plac-
15   ing routers at each site and using leased lines to interconnect them. A router that has a wide-area connection to another router may be called a "backbone router." The "back-bone network" is the set of backbone routers and their interconnections.

If every backbone router is connected to every other backbone router, the back-bone network is said to be "fully meshed." In a fully meshed backbone network, data
20   that travel from one site to another go through the backbone router at an origin site ("in-gress router"), travel over the leased line to the backbone router at the target site ("egress

1

router"), and then enter the target site. More commonly, though, the backbone network is not fully meshed; a router is connected to only a small number of others (three or four). In such a sparse topology, the ingress and egress routers may not be directly connected. In this case, data may have to pass through several additional, "transit" routers on the way from ingress to egress.

In a private network like this, the design and operation of the backbone network is the responsibility of the enterprise. A routing algorithm must run in the backbone routers, enabling them to tell each other the addresses of the destinations to which they can respectively afford access.

It is worth noting that a leased line is not actually a piece of wire going from one site to another. It is really a circuit through some circuit-switching network. But this is of no import to the enterprise network manager, to whom those circuits can be considered simple unstructured pipes. Conversely, although the telephone network itself requires considerable management, the telephone-network managers do not need to know any-thing about the enterprise backbone network; to them, the telephone network just pro-vides point-to-point connections. They do not need to know what role these connections might be playing in an enterprise data network.

We may say that the enterprise network is "overlaid" on top of the telephone net-work. The enterprise network can be called the "higher layer" network, the telephone network the "lower layer" network. Both networks exist, but each is transparent to the other. The enterprise's backbone routers exchange routing information with each other, but the telephone switches do not store or process that routing information. That is, backbone routers are "routing peers" of each other, but they are not routing peers of the telephone switches. This way of building a higher-layer network on top of a lower-layer network is called the "overlay model."

2. Virtual Private Networks

Wide-area enterprise networks are now more likely to be built on top of frame-relay and ATM networks than on top of circuit-switched (telephone) networks. Whereas a telephone network really provides circuits between backbone routers, a frame-relay or

ATM network provides "virtual circuits" between backbone routers. But this changes nothing as far as the enterprise's routing task is concerned; the overlay model still applies even though the lower-layer network is now a frame-relay or ATM network rather than a circuit-switched one, i.e., even though virtual rather than fixed circuits make the point-to-

5 point connections between backbone routers. The two networks are still transparent to each other. The enterprise network manager still has a wide-area backbone to design and operate. However, because the circuits are "virtual," this is usually called a "virtual pri-vate network" (VPN) instead of a "private network."

Since the two networks are transparent to each other in the overlay model, that

10 model is distinguished by the fact that the enterprise's backbone routers do not share with the (service provider's) frame-relay or ATM switches the routing information that they must share with each other. This causes inefficiency when the enterprise's backbone routers are not fully meshed. In such networks, some packets go from the ingress router through one or more transit routers before they reach the egress router. At each one of

15 these "hops," the packet leaves the frame-relay or ATM network and then enters it again. This is sub-optimal—there is little value in having a packet go in and out of the frame-relay or ATM network multiple times.

This problem can be avoided by making the enterprise backbone fully meshed, but that causes problems of its own. The number of virtual circuits the enterprise has to

20 pay the service provider for to make the network fully meshed grows as the square of the number of backbone routers. Apart from the cost, routing algorithms tend to scale poorly as the number of direct connections between routers grows. This causes additional prob-lems.

The overlay model also tends to result in extra traffic when multicast is in use. It

25 is usually impractical or undesirable for the "lower layer" network to do the necessary packet replication, so all packet replication must be done in the "higher layer" network, even if a number of replicated packets must then follow the same "lower layer" path up to a point.

3. The Peer Model

Since these considerations all impose upon the resources of an enterprise for which communications is not necessarily a core competence, a service provider ("SP") can afford its customers greater value if it absorbs the task of designing and operating the

5   backbone. More specifically, the SP should so organize and operate the backbone that, from the point of view of a particular site administrator, every enterprise network address not located at a given site is reachable through the SP's backbone network. How the SP's backbone decides to route the traffic is the SP's concern, not that of the customer enterprise. So the customer enterprise does not really need to maintain a backbone router at

10   each site; it just needs a router that attaches to one of the SP's backbone routers. As will become apparent, providing such an organization involves abandoning the overlay model for a different model. For reasons that will be set forth below, we call the new model the "peer model."

Terminology:

15   *C-network*: the enterprise network, consisting of C-routers, which are maintained and operated by the enterprise.

P-network: the SP network, consisting of P-routers, which the SP maintains and operates.

CE-router: an "edge router" in the C-network, i.e., a C-router that attaches

20   directly to a P-router and is a routing peer of the P-router.

PE-router: an "edge router" in the P-network, i.e., a P-router that attaches directly to a C-router and is a routing peer of the C-router.

If a P-router is not a PE-router, i.e., not an edge router, it is a *transit router*. The concept of edge and transit routers is relative to specific VPNs. If a given one of the SP's

25   routers receives a given VPN's traffic from and forwards it to only others of the SP's routers, the given router is a transit router vis-à-vis the given VPN. Yet that same router may receive another VPN's traffic from and/or forward it to one of that other VPN's edge routers, in which case the given SP router is an edge router from the other VPN's point of view.

In the conventional peer model, where "virtual routers" (i.e., one instance of the routing algorithm per VPN) are used, all C-routers within the same VPN are routing peers of each other. But two C-routers will be routing adjacencies of each other only if they are at the same site. Each site has at least one CE router, each of which is directly

5   attached to at least one PE router, which is its routing peer. Since CE routers do not exchange routing information with each other, there is no virtual backbone for the enterprise to manage, and there is never any need for data to travel through transit CE routers. Data go from the ingress CE router through a sequence of P-routers to the egress CE router. So the resultant routing is optimal. These clear customer benefits have led certain

10   SPs to adopt the peer model.

The conventional peer-model approach also enables the SP to solve certain problems that arise from using a common backbone network for more than one client. One of these is address duplication. Although there is an international assigned-number authority from which unique addresses can be obtained, many enterprise networks simply as-

15   sign their private-network addresses themselves. So their addresses are unique only within the particular enterprise: they may duplicate addresses that another customer enterprise uses. An SP trying to use, say, an Internet-Protocol ("IP") backbone as the backbone for different enterprise networks having overlapping address spaces needs to provide its P-routers with a way of identifying and selecting a route to the one of potentially

20   many same-address destinations to which it should forward a packet.

So the SP makes use of a "virtual router." When a PE router receives a packet received from a CE router, the PE router "tags" the packet with an indication of the C-network where it originated. It then bases its determination of what router to forward the packet to not only on the packet's destination address but also on the identity of the

25   originating C-network. At each subsequent hop, the router looks up the packet's destination address in the forwarding table specific to the C-network that the tag designates.

This also solves another multiple-customer problem, that of the access control. If an enterprise buys network-backbone service from an Internet SP, it wants some assurance that its network receives only packets that originated in its own network. It also

30   wants to be sure that packets originating in its network do not leave the enterprise net-

work by accident. Of course, two enterprises might want to be able to communicate directly, or to communicate over the Internet. But they want such communication to occur only through "firewalls." By using the virtual router, the SP solves this problem, too.

SUMMARY OF THE INVENTION

5   We have devised a way for an SP to provide its customers the peer model's advantages at costs considerably lower than those that the conventional virtual-router approach exacts. The way in which we do this enables transit P-routers to base their routing decisions for VPN-destined packets on packet fields that the transit routers interpret without reliance on VPN-specific routing information.

10   To appreciate the resultant improvement, one must recognize that one of the main problems in large IP backbones is the quantity of resources (memory, processing, bandwidth) needed to maintain the routing information. If an SP now adds parallel routing information for, say, a hundred C-networks, the virtual-router approach requires that each physical P-router act as a hundred "virtual" C-routers. That is, each P-router runs one-
15 hundred routing algorithms (one for each C-network), maintains one-hundred forwarding tables, etc. Clearly, this greatly exacerbates the existing information-maintenance problem.

       Indeed, because of poor topological matching, the problem is even worse than it appears at first glance. SPs generally try to assign addresses in a topologically meaning-
20 ful way. That is, the address a system has should be related to where it attaches to the SP's network. This sort of addressing scheme allows routing information to be aggregated; if addresses of links that are topologically close tend to have the same high-order bits, for instance, a single routing-table entry can be used for a large number of relatively close destinations. Such aggregation reduces the routing load on the P-routers.

25   But many customer enterprises' networks have addressing schemes that do not map well to the SP's backbone topology. So passing enterprise routing information into the P-network, as a peer-model organization does by definition, exacerbates routing-information overload.

And the virtual-router approach introduces a problem of its own. The C-network's interior-routing algorithm is now running both in C-routers and in P-routers. This blurs the administrative boundaries. If the routing algorithm fails, two different administrations must work together to troubleshoot it. This is in direct contradiction to a lesson of nearly all networking experience: interior-routing algorithms must be confined to a single administrative domain.

We greatly reduce these problems by a judicious use of packet tagging. PE routers in systems that implement the present invention's teachings still use VPN-specific routing information in making their routing decisions, so they still must maintain tables of such VPN-specific information. But they relieve transit routers of the need to do so. When a PE router receives a packet from a customer enterprise, it provides the packet with an internal-routing field, which a transit router can interpret as specifying the egress PE router that will forward the packet into the remotely located part of the customer-enterprise network. For example, the internal-routing field may include a tag that the transit router can interpret as specifying the next router on the way to that egress PE router. Consequently, transit routers need not concern themselves with routes to locations in the customer enterprises' networks. Yet the egress PE router can still determine the particular VPN into which it is to forward the packet, because the first PE router, which can identify the VPN by the CE router from which it received the packet, so forms the internal-routing field's contents that it additionally specifies the target VPN. For example, the internal-routing field can include a second tag, one that the egress router can interpret as identifying the CE router that affords access to the addressed host in the proper VPN.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention description below refers to the accompanying drawings, of which:

Fig. 1 is a topological diagram of a VPN and a tagging sequence that its routers employ;

Fig. 2 is a diagram that illustrates the format of a tagged packet;

7

Fig. 3 is a diagram of the environment and format of a tag-distribution-protocol protocol data unit;

Fig. 4 is a diagram that illustrates the format of a conventional Border Gateway Protocol protocol data unit and its environment;

Fig. 5 is a diagram that illustrates the format and environment of a Border Gateway Protocol protocol data unit used to distribute VPN-distinguishing reachability information and tags; and

Fig. 6 is a diagram that illustrates the format of another conventional Border Gateway Protocol protocol data unit and its environment;

Fig. 7 is a topological diagram of a VPN that employs ATM switches in implementing the present invention's teachings;

Fig. 8 is a diagram of an ATM frame used in the Fig. 7 embodiment;

Fig. 9 is a topological diagram used to illustrate inter-VPN communication;

Fig. 10 is diagram that illustrates the format of a label-stack entry that may be used in implementing the present invention's teachings;

Fig. 11 is a state diagram of that describes aspects of the Tag Distribution Protocol;

Fig. 12 is a diagram tha depicts the general format of the Tag Distribution Protocol protocol data unit ("a TCP PDU");

Fig. 13 is a diagram that illustrates the format of a time-length-value field;

Fig. 14 is a diagram that illustrates a particular format of a TDP PDU;

Fig. 15 is a diagram that depicts the format of a TDP_PIE_OPEN message;

Fig. 16 is a diagram that depicts the format of protocol information elements that carry optional parameters;

Fig. 17 is a diagram that depicts the format of a message field used to represent virtual-circuit-indicator ranges;

Fig. 18 is a diagram that depicts the format of a message field used to represent tag values;

Fig. 19 is a diagram that depicts the format of a field used to specify a particular type of tag binding;

Fig. 20 is a diagram that depicts a different format used for that purpose;

Fig. 21 is a diagram that depictts yet another format used for that purpose;

Fig. 22 depicts a TDP_PIE_REQUEST_BIND protocol information element ("PIE");

Fig. 23 depicts the format of an ALIST entry for ALIST_TYPE = 1;

Fig. 24 depicts the format of an ALIST entry for ALIST_TYPE = 2;

Fig. 25 depicts the format of a TDP_PIE_NOTIFICATION PIE used for error reporting:

Fig. 26 depicts the format of an error notification that results from receipt of a binding request containing a hop-count value that exceeds the maximum hop count;

Fig. 27 depicts the format of value field of a NO_ROUTE parameter;

Fig. 28 depicts the format of the TDP_PIE_WITHDRAW_BIND PIE;

Fig. 29 depicts the format of the TDP_PIE_RELEASE_BIND PIE;

Fig. 30 depicts the format of the TDP_PIE_KEEP_ALIVE PIE;

Fig. 31 depicts the format of the TDP_PIE_NOTIFICATION PIE;

Fig. 32 illustrates the use of such a PIE's RETURNED_PDU parameter;

Fig. 33 illustrates the use of such a PIE's CLOSING parameter;

Fig. 34 depicts the format of a TDP_PIE_OPEN PIE;

Fig. 35 depicts the format of a TDP_PIE_REQUEST_BIND PIE; and

Fig. 36 illustrates the encoding of the MP_REACH_NLRI attribute.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

Overview

Before we describe an embodiment of the invention in detail, we will employ Fig. 1 to present a brief overview of its operation.

Fig. 1 depicts a very simplified topology for illustrating an SP's connections between two parts of a customer enterprise C's VPN. Two of the enterprise's edge routers CE1 and CE2 are located remotely from each other, and the customer enterprise has contracted with the SP to provide connections between the customer's routers such as CE1

and CE2 to form a VPN V. Among the SP's resources are edge routers PE1 and PE2 and further, transit routers P1 and P2 that together form a path between CE1 and CE2.

Consider a packet that a router CE2 receives from a location (not shown) in VPN V, and suppose that the contents D1 of the packet's destination-address field is the address of a system in VPN V at CE1's location. We assume that CE2 has interfaces over which it could potentially have forwarded the packet to routers, not shown in the drawing, to which it is directly linked but that it concludes by consulting stored routing information that it should forward the packet over its interface to edge router PE2 in the SP.

We also assume that the SP network has another customer for which it uses those same resources to implement a different VPN, W, that also includes a (differently located) host having the same address D1. From the fact that PE2 has received the packet over its link with CE2, which is part of V rather than W, PE2 can tell which D1-addressed system should receive the packet. The VPNs that the SP cooperates with its customers to implement follow the peer model, so PE2 contains customer-network topological information that the customers have "leaked" to it. It stores this information in a separate routing table for each customer VPN to which PE2 is directly connected, so it can disambiguate the otherwise ambiguous address D1. From this information, PE2 knows that PE1 is the SP edge router to which it should direct the packet in order to reach the D1-addressed system in VPN V.

Now, the goal is to have that other edge router, PE1, forward the packet to CE1 so that the packet will reach the D1-addressed location in VPN V rather that the one in VPN W, to which PE1 may also be able to forward packets. Therefore, PE2 needs to include in the forwarded packet some indication that the intended D1-addressed host is the one in VPN V. In accordance with the invention, though, this should be done without requiring that transit routers P1 and P2 also maintain the VPN-specific information that the edge routers store.

PE2 achieves this by adding to the packet an internal-routing field that in the illustrated embodiment includes two constituent fields, namely, an egress-router field and an egress-channel field. The egress-router field takes the form of a tag that P2 can map to

the next hop in the route to the egress edge router PE1, upon which the transit routers can base their routing decisions without requiring knowledge of the VPN involved. The egress-channel field takes the form of a tag that PE1 can interpret as specifying its interface with CE1 or as otherwise representing the channel that links it to VPN V.

5   Note that the goal of avoiding VPN-specific forwarding information could be achieved, though to a lesser degree, by having the internal-routing field include only an egress-channel field, not an egress-router field. The transit routers would then be basing their routing decisions on fields that in a sense do designate particular VPNs, but only because a given channel may lead only to nodes in a particular VPN. The transit routers

10   still would not need to store information concerning locations in any of the customer sites.

But we prefer to use both an egress-router field and and egress-channel field. Specifically, PE2 "tags" the packet with two tags T2 and T3. As will be explained in detail below, P2 has arranged with its neighbors, including PE2, to tag with T2 any pack-

15   ets sent to P2 for forwarding along a route in which the SP edge router is PE1. T3 is a tag with which PE1 has arranged for the other edge routers to tag packets destined for certain VPN V locations if PE1 is the egress router.

To describe one way to tag a packet, we begin with Fig. 2's first row, which illustrates an exemplary link-level-protocol format. Different link-level protocols may be

20   employed on different links. Examples of such protocols are the IEEE 802 protocol family and the point-to-point protocol (PPP) specified in the Internet community's Requests for Comments ("RFCs") 1331 and 1332. Similar to the former is the Ethernet protocol. If the links connecting CE2 to PE2 and PE2 to P2 are Ethernet links, the link-layer frame that CE2 sends to PE2 takes the form that Fig. 2's top row depicts. Specifically, it consist

25   of a link-level payload encapsulated by an Ethernet header and trailer. The Ethernet trailer consists of a cyclic-redundancy-code (CRC) field used for error detection. The Ethernet header includes destination-address and source-address fields, which respectively contain the link-level ("hardware") addresses of PE2's and CE2's interfaces to that link, and it also includes a type field used for demultiplexing the link packet's contents.

30   In this case, the code represents the Internet Protocol (IP): the receiving router should in-

terpret the contents as an IP "datagram" (as the IP protocol data unit is called), consisting of the IP header and IP data. (Of course, the payload could be a protocol data unit of some other network-level protocol, such as IPX or Appletalk.) Routers generally use network-protocol information to forward packets from one link to another along an inter-

5   network path from the source interface to the ultimate destination interface.

Fig. 2's second row depicts the corresponding link-layer frame after PE2 has added T2 and T3. The Ethernet header and trailer take the same form as before. (For the sake of discussion, we assume that the link-level protocol is the same on the new link, although most embodiments will not exhibit such protocol uniformity.) Since the link-

10   level source and destination are different, of course, the corresponding header fields' contents differ from those in the CE2-to-PE2 frame, and the CRC field contents, having been calculated from different frame contents, are different, too. But the difference most relevant to the present discussion is the type-field difference. Even though the frame does include an IP datagram, the type field does not contain the IP-indicating code. In-

15   stead, the code that it contains tells P2's interface that the frame's contents should be interpreted as a tagged packet.

This means that the four bytes immediately following the link-level header should be interpreted as an entry in a "tag stack," whose format Fig. 2's third row illustrates. Specifically, the first twenty bits should be interpreted as the tag, and the twenty-fourth,

20   bottom-of-stack-indicator bit S tells whether the packet contains any more tag-stack entries. (The section below entitled "Internet Draft: Label Stack Encodings" contains a thorough description of the manner in which a tag-switching router can use the various fields, so we will not discuss the other, COS and TTL fields here.) In the example the tag field contains the "top" tag value T2, while the S bit is zero, indicating that this is not the

25   bottom tag-stack entry. Therefore, P2 should interpret the next four bytes as a tag-stack entry, too. In the example, that entry contains a tag value of T3 and an indication that it is the bottom stack entry.

We now return to Fig. 1 and assume that PE2 has just sent P2 a packet thus tagged. Since T2 is a tag that P2 has arranged to have PE2 attach to packets that should

30   follow routes in which PE1 is the egress router, P2 knows to forward that packet to the

neighbor, P1, to which it sends PE1-directed traffic. (Again, P2 must make a routing decision because we assume that it additionally has direct links to other routers.) Note that P2 is able to make this decision without having had to maintain separate routing information for the VPN to which the packet is ultimately destined.

5   When P2 forwards the packet to P1, it replaces tag T2 with a new tag, T1, which P1 has asked its neighbors to attach to any packets that should be sent though PE1-egress routes, and P1 similarly makes its routing decision without having had to maintain separate routing information for the destination VPN. P1's stored routing information tells it to remove a tag rather than replace it, so it does so before forwarding the packet to PE1.

10   From tag T3, PE1 knows that it should forward the packet to the edge router CE1 that affords access to the D1-addressed location in VPN V. So PE1 forwards the packet to CE1 after removing tag T3. Since CE1 is concerned only with destination addresses in its own VPN, it is able to base its routing decision on D1 alone.

General Routing Features

15   Having now considered the illustrated embodiment's overall operation, we turn to a review of certain network-operation concepts that will provide a foundation for a more-detailed discussion of the operation described in the above overview. In a typical implementation, router circuitry for performing functions described below will be provided as communications hardware operated by one or more processors software-configured to

20   perform the described operations. Those skilled in the art will recognize that such an approach is usually the most practical, because software configuration of a general-purpose processor enables a relatively small amount of hardware to serve as circuitry for performing many different functions concurrently. But the present invention can instead be implemented in any circuitry that performs the functions described.

25   **1. The FIB**

In conventional IP forwarding, each router maintains a table, sometimes called the "Forwarding Information Base" (FIB), that it uses to map from "address prefixes" to "next hops." A router that receives a packet whose destination address begins with a

given address prefix employs the next-hop entry as described below to determine the direction in which to forward the packet.

The manner in which the FIB is constructed is not critical to the present invention. In principle, a system administrator can provide it manually. More typically, routers build such tables automatically by employing routing algorithms to share topological information. But regardless of how the FIB is constructed, a conventional router R executes the following procedure (in principle) to find the next hop for a particular packet:

- It searches the FIB for longest address prefix that matches the IP (or other network-level) address in the packet's network-level destination-address field.

- It fetches the next-hop IP address, N, that corresponds to that address prefix.

- If N is the address of a router to which R is directly connected (i.e., if there are no routers between R and the next hop), then the procedure ends, and R forwards the packet over its link to the router whose address is N.

- If N is not the address of a router to which R is directly connected, then R performs a *recursive* lookup. That is, it searches the FIB for the longest address prefix that matches N, fetches the corresponding next-hop IP address N2, determines whether N2 is directly connected, etc. The recursion ends when R finds a next hop directly connected to it, and it R forwards the packet over its link to the router whose interface has that address.

In practice, as those skilled in the art will recognize, the FIB will have been preprocessed to eliminate the need to perform the recursion during actual packet processing. To avoid complicating the discussion unnecessarily, though, we omit a description of such conventional preprocessing.

A normal Internet router maintains only one FIB table. But routers in a provider of connections for many enterprises' peer-model VPNs need different tables for different VPNs, because a router may need to distinguish between potentially identical prefixes in different VPNs. (Each SP router also needs to maintain a general, i.e., non-VPN-specific, FIB. Unless explicitly stated otherwise, references below to the FIB mean the general FIB.) In accordance with the present invention's teachings, though, transit routers, i.e.,

14

routers that are not directly attached to customer's VPN, do not need to maintain VPN-specific FIBs. (We consider a PE router to be "directly attached" to a particular VPN if it is directly attached to a CE router in that VPN.) And an edge router such as PE1 or PE2 needs to maintain, in addition to a general FIB, a separate FIB only for each VPN to

5   which it is connected directly. The reason why this is so will become apparent as the description proceeds.

In the illustrated embodiment, each FIB entry actually differs somewhat from that described above, because the illustrated embodiment uses "tag switching." When data-transmission speeds become high and network sizes become large, searching for longest

10   matches to the packet's ultimate-destination address becomes onerous. So proposals have been made to reduce this burden by "tagging" the packets.

A *tag* is a field that routers use to make routing decisions. Unlike a network-level address, though, a tag is a true (unique) index to a given router's routing table, whereas the network (e.g., IP) address in the destination field of a packet's header is merely an

15   invitation to a router to find the address prefix that constitutes the best match. By reducing the need for best-match searches, conventional tagging reduces a router's processing burden. And we use tagging in such a way as additionally to reduce routers' storage burdens, as will become apparent after a discussion of further tag-switching and other features.

20   One way to implement tag switching is to have routers tell their neighbors the tags they want to see in the packets that they receive. Specifically, a given router may decide to associate a particular tag with ("bind a particular tag to") a particular address prefix. If so, it tells its neighbor routers that, when they forward it a packet destined for an address having that prefix, they should attach the specified tag so that the given router can go

25   straight to the right table entry without having to do a best-match search. (Although the illustrated embodiment bases tagging on address prefixes, other embodiments may base it on some other packet attribute that is relevant to routing.)

When tag switching is used, the forwarding table does not merely map an address prefix to a next-hop IP address; it maps the address prefix to an ordered pair whose first

element is a next-hop IP address and whose second element is a tag-stack operation. That is, an FIB next-hop entry contains both a next-hop IP address and a tag-stack operation.

Initially, we need to consider only two tag-stack operations:

- No op.
5
- Push a specified tag value onto the stack.

The "no op" value is the default tag-stack-operation entry. As will be explained below, neighbors' requests may result in that entry's being modified to contain a push operation.

When router R receives an untagged packet, it finds the longest address-prefix
10 match to R's destination IP address, and it fetches the corresponding next-hop entry. If that next-hop entry's tag operation is "push a specified tag value onto the stack," it pushes the specified tag value onto the tag stack that the packet includes. If it is necessary for R to perform a recursive lookup, it searches for another next-hop entry. If that next-hop entry also has a "push a specified tag value onto the stack" operation in it, that
15 specified value is also pushed. If the recursion ends as a result of the second lookup, then two tag values may have been pushed onto the tag stack.

When the recursion ends (or if there is no recursion), R knows which of its directly connected neighbors is the next hop for the packet. It then transmits the packet to that next hop, using whatever data-link protocol is necessary in order to reach that next
20 hop.

2. The TIB

When a router R uses tag switching, it fetches next-hop information in response to a tag, so it uses a routing table separate from the FIB, from which it fetches next-hop information in response to a destination address. This separate table is sometimes called
25 the Tag Information Base (TIB). The TIB next-hop entries contain a next-hop IP address and a tag-stack operation. For our purposes, we need consider only three tag-stack operations:

- remove the tag stack's last-added ("top") value ("pop the stack");

- replace the top tag-stack value with a specified value; and

- discard the packet.

When router R receives a tagged packet, it uses the packet's top tag as an index into the TIB and fetches the indicated entry. (Those skilled in the art will recognize that security requirements, local-link constraints, or other considerations may in some cases necessitate that the index into the TIB actually consist of both the incoming packet's tag and the interface on which it arrived, but the principle is best explained without complicating the discussion with those details.) In accordance with the fetched TIB entry, it either replaces the tag with a different value or pops the tag stack.

If the TIB entry's next-hop field is the address of one of R's directly connected neighbors, R uses the appropriate data-link protocol to send the tagged packet to that neighbor. If the next hop specified in the TIB entry is not a directly connected neighbor, on the other hand, then R (again, in principle) performs a recursive lookup by finding the FIB entry that corresponds to that address. (The FIB is used since this part of the search is based on an address, not a tag.) Then processing proceeds as described in "The FIB" above.

3. How Interior Routing Algorithms Modify the FIB and the TIB

As was stated above, the present invention does not require any particular mechanism for providing the contents of the FIB and the TIB. But considering one such mechanism, namely, routing protocols, helps one appreciate those contents' purpose. The types of protocols that it uses can be divided into interior gateway protocols (IGPs), exterior gateway protocols (EGPs), and tag-distribution protocols (TDPs). Routers in an internetworking domain under single administration use IGPs to share topological information about that domain. Routers use EGPs to share extra-domain topological information. They use TDPs to distribute tags.

Typically, every router runs an IGP. Examples of such protocols are OSPF, EIGRP, and IS-IS. From time to time, a router sends to its same-domain neighbor routers IGP messages that "advertise" destinations to which it accords direct access. The neighbors in turn forward the messages to their neighbors. In some protocols the forwarding

routers modify the messages in such a way that a message tells what route it took to reach the recipient, or at least how long the route was. In any case, the recipient thereby amasses topological information and decides on the basis of that information whether to enter into its FIB as the next hop to the advertised destinations the address of the router

5   that forwarded it the message. So FIB entries that an IGP creates are always non-recursive: the next hop is always a directly connected neighbor.

The customer-enterprise routers may also use an IGP. Although the drawing does not show them, the customer enterprise would typically also have further routers at the same sites as CE1 and CE2, and those routers may use an IGP. But the customer enter-

10   prise's nodes that have access to each other only through the provider network do not use an IGP to exchange routing information with each other, so the routers at, for instance, CE1's site use an IGP only for routing-information exchange with other routers at the same site (or other sites to which there is customer-managed access), not for such exchange with routers at CE2's site.

15   When IGP maps address prefix X to next hop N, it may modify both the FIB and the TIB. The FIB modifications are as follows:

- If the FIB already contains an entry that maps X to a next hop, and the next hop is N, then no change is made.
- If the FIB does not already contain an entry that maps X to any next hop,

20   or if the FIB already contains an entry that maps X to a next hop other than N, then IGP inserts an entry that maps X to N and removes any entry that maps X to a different router. In a tag-switching routine, the IGP process then determines whether N has sent R a message that binds X to some tag value T. If not, the FIB entry is inserted with the tag-stack operation

25   "no op." Otherwise, the FIB entry is inserted with the tag-stack operation "push T onto the stack."

The TIB modifications are as follows:

- If no FIB modification has been made, then no TIB modification is made, either.

18

- If an FIB modification has been made, then R determines whether it has told any of its directly connected neighbors to tag X-destined packets with some tag value T. If not, it makes no TIB modifications. Otherwise, it looks up the TIB entry that corresponds to T.

- If there is no corresponding TIB entry, R inserts one for tag T having a next-hop entry of N. If there is a corresponding TIB entry, it replaces the next-hop entry with N.

- R then determines whether N has asked it to tag X-destined tags with some tag value T2. If not, the tag-stack operation is "discard the packet." Otherwise:

  * If N's requested "tag" T2 for X-destined packets is a actually a distinguished tag value that means "pop the tag stack," then N has not really asked that R place a tag on such packets but instead has asked that it merely remove one already in the packet. So TIB entry's tag-stack operation is "pop the tag stack."

  * Otherwise, the TIB entry's tag-stack operation is "replace the packet's top tag-stack value with T2."

A distinguished value of "next hop" that may exist in both the FIB and the TIB is "me." This means that a packet has reached its final hop, and is delivered to local software rather than forwarded over a data link to a next hop.

4. Edge Routers and the IGP

Now, it was stated above that IGP speakers periodically advertise address ranges to which they afford direct access. If P1 is on a subnet in which all hosts' addresses start with 192.3.45, for instance, it will advertise this prefix, and every IGP speaker in the SP network will have an entry for that prefix in its FIB. Therefore, if PE1 has an interface on the same subnet, say with an address of 192.3.45.12, then those IGP speakers will be able to determine how to reach PE1. But it will become apparent as the description proceeds that, in order to assign certain tags, the illustrated embodiment requires each SP router additionally to have PE1's full address as a prefix in its FIB. And, in general, each

19

SP router should have such a "host route" for every PE-router. (A *host route* is one whose prefix is the length of a complete IP address and thus corresponds to only one host.) So edge routers in the illustrated embodiment advertise not only the address ranges to which they have access but also their own complete addresses. (Actually, as will

5 shortly be explained, the edge routers are also "BGP speakers," which would conventionally advertise their host routes in IGP anyway.)

5. How BGP Modifies the TIB and the FIB

It was mentioned above that IGPs are used for propagating routing information among routers connected by routes within a commonly administered domain. In such a

10 domain, the assumption is that routers are generally to cooperate in routing any received packets and that they will accumulate routing information from all sources within that domain. But a domain administered by one entity may additionally be connected to domains administered by others. For such connections, a given domain may choose to be selective about what traffic it will forward and which of its resources it will make avail-

15 able for that purpose. Additionally, it typically is not practical to accumulate routing information from all routers in every other domain, even if the other domains were inclined to supply it, so inter-domain topology-information sharing calls for some selectivity.

This is not something to which IGPs are well suited. For communicating information of that type, therefore, routers involved in communication among such "autono-

20 mous systems," as they are called, use external routing protocols, such as External Gateway Protocol (EGP). For the sake of concreteness, we assume here that the external routing protocol used here is the one specified in RFC 1654 and referred to as the Border Gateway Protocol (BGP).

In BGP, the type of message used to advertise a route is called an "update" mes-

25 sage. In a conventional, non-tag-switching BGP implementation, an update message contains an address prefix, a "BGP next hop," and an AS Path, which lists the autonomous systems traversed in reaching the advertised destinations. With tag switching, this is modified to add a tag to each address prefix.

When a router R receives a BGP update message for address prefix X from a BGP peer R2, R runs the BGP decision process. Policies that the BGP process implements may or may not result in R's installation of R2's route to X. But if they do, then:

- If the FIB does not already contain an entry that maps X to a next hop, or if it contains an entry that maps X to a next hop other than the one specified in R2's BGP update message, then R adds an entry that maps X to the specified next hop, and it removes any previous entry for X. This next hop will not in general be a directly connected neighbor of R, so the FIB entry may be a recursive one. (In the cases in which we are interested, R2 will specify itself as the BGP next hop, in which case the FIB entry will map X to R2.) If R2's BGP Update message specified tag value T for address prefix X, then the tag-stack operation in the FIB entry is "push T onto the tag stack." Otherwise, the tag-stack operation is "no op."

- If the FIB already contains an entry that maps X to a next hop, and the next hop is the same as the one specified in R2's BGP Update message, then the FIB entry's next-hop field is left unchanged. If R2's BGP Update message specified tag value T for address prefix X, then the FIB entry's tag-stack operation is changed (if necessary) to be "push T onto the tag stack." If R2's BGP Update message specifies no tag value for X, then the tag stack operation in the FIB entry is changed (if necessary) to "no op."

6. The Decision to Distribute a Tag Binding

The preceding discussion concerned what happens when a router has asked another router to associate a tag with a prefix. We now describe the circumstances under which a router makes such a request.

In most tag-switching proposals, a router is allowed to bind a tag to an address prefix if the router's FIB table includes an entry that corresponds to that address prefix. In the illustrated embodiment, if the FIB-entry "prefix" is the complete address ("host route") of a router in the SP's network, then binding a tag to that prefix is not only permitted but required.

If X is the (thirty-two-bit) address of the router R itself, then the tag value that R binds to X is the distinguished value that means "pop the tag stack."

When a tag T is bound to an address prefix X, and the FIB entry for X was inserted as a result of running the IGP, R will distribute the tag binding to its directly connected neighbors by using a tag-distribution protocol that will be described below.

When a tag T is bound to an address prefix X, and the FIB entry for X was inserted as a result of running BGP, R will use BGP to distribute the tag binding, in a manner that will be described below, to any BGP peer to which it distributes the route to X.

If router R binds to an address prefix X a tag T other than the distinguished value that means "pop the tag stack," then R also creates a T-indexed TIB entry in its own TIB table.

The TIB entry is created as follows.

- Suppose that R is a PE router, and address prefix X is one for which the next hop is a directly attached CE router. (As will be explained below, the prefix value will have been enhanced to distinguish X in CE's VPN from X in others'.) Then the TIB entry will specify the CE router as the next hop, and its tag-stack-operation entry will be "pop the tag stack."

- Suppose that the FIB entry corresponding to X specifies a next hop of N and a tag-stack operation of "push value T2 onto the stack." Then the TIB entry will give N as the next hop and "replace the value at the top of the stack with T" as the tag-stack operation.

- Suppose that the FIB entry corresponding to X specifies a next hop of N and a tag-stack operation of "no op." Then the TIB entry will specify a next hop of N, and a tag-stack operation of "discard the packet."

Detailed Example

We now have enough background to describe in detail the way in which the illustrated embodiment performs the operations mentioned briefly in connection with Fig. 1 For this purpose, we return to Fig. 1.

22

All of Fig. 1's P routers (PE1, PE2, P1, and P2) participate in a common IGP. CE1 and CE2 do not participate in this IGP. CE1, PE1, CE2, and PE2 are BGP speakers. CE1 has an External BGP (EBGP) connection to PE1, PE1 has an Internal BGP (IBGP) connection to PE2, and PE2 has an External BGP connection to CE2. (As those skilled in the art are aware, the way in which a BGP speaker reacts to BGP messages originating in its own autonomous system differ from the way in which it responds to BGP messages that originate in a different autonomous system. The BGP session is commonly referred to as "internal" in the former case and "external" in the latter.)

1. FIB Entries That IGP Creates

Since PE1 is an edge router, it exports its own thirty-two-bit address into the P-network's IGP. As a result:

- PE2 has an FIB entry that maps PE1 to a next-hop value of P2. Since P2 is directly connected to PE2, this entry is non-recursive.

- P2 has an FIB entry that maps PE1 to a next-hop value of P1. Since P1 is directly connected to P2, this entry is non-recursive.

- P1 has a FIB entry that maps PE1 to a next hop value of PE1. Since PE1 is directly connected to P1, this entry is non-recursive.

- PE1 has a FIB entry that maps PE1 to a next hop value of "me."

2. TDP Messages; TIB Entries Created As a Result of TDP Processing

As was mentioned above, the illustrated embodiment requires that each of the SP's routers construct a TIB by assigning tags to all of the prefixes for which its FIB has entries and that it ask its neighbors to use those tags in forwarding data packets to it. A mechanism that they can use to make those requests is a tag-distribution protocol (TDP). The section below entitled "Internet Draft: Tag Distribution Protocol" describes that protocol in detail. Here we only digress briefly to mention certain salient features.

TDP is a two-party protocol. It requires a connection-oriented transport layer that provides guaranteed sequential delivery. Fig. 3's second row therefore depicts TDP's protocol data units (PDUs) as being carried in a data stream delivered by the well-known

Transport Control Protocol (TCP) whose segments are delivered in Internet Protocol (IP) datagrams whose format Fig. 3's first row depicts. (That row omits the link-level-protocol header and trailer fields that usually encapsulate the IP datagram for transmission between hosts on the same link.)

5   The IP datagram begins with a header that includes various types of information such as the datagram's length, the network address of the destination host interface, and a code for the next-higher-level protocol in accordance with which the destination host should interpret the datagram's payload. In the illustrated example, that protocol is TCP, which handles matters such as ensuring that data have been received reliably. As the

10   drawing illustrates, the destination host's TCP process interprets the first part of the IP field as a header used in carrying out these TCP functions. In particular, that header includes a field that specifies the "port" application that is to receive the TCP segment's remainder, payload portion. In the case under consideration, the port field indicates that the host's TDP application is to receive it.

15   Concatenation of TCP-segment payloads results in a data stream that contains the TDP PDUs.

A TDP PDU begins with a fixed-length four-field header. The header's two-byte version field gives the number of the TDP version that the sender is using. The two-byte length field gives the length in bytes of the remainder of the PDU; i.e., it gives the total

20   PDU length minus four.

As will be explained shortly, TDP communications occur in sessions, of which a given router can be conducting more than one at a time. The first four bytes of the six-byte TDP ID field encode an IP address assigned to the router that started the TDP session, and the TDP ID field's last two bytes identify the particular session.

25   A two-byte field reserved for further enhancements completes the header, and the remainder of the PDU comprises one or more protocol information elements (PIEs), which take the type-length-value format that Fig. 3's third row illustrates.

Each PIE's type field specifies its purpose, while its length field gives the length of its value field. Various PIE types have housekeeping purposes, such as instituting a

TDP session between two routers, negotiating protocol versions, providing error notifications, and keeping the session alive. (If a router does not receive a same-session communication within a certain timeout period, it ends the session and discards the tags installed during the session.) But the protocol's main mission, i.e., distributing tag bindings, is

5   carried out by PIEs of the TDP_PIE_BIND type, for which the type field's contents are 0200_{16}.

Fig. 3's fourth row depicts this PIE type's value segment. In that segment the request-ID field is zero unless the PIE is being sent in response to a request from the other session participant, in which case that field's request ID matches that of the request.

10   (Such a request would have been sent as another PIE type.) The AFAM (Address Family Numbers) field is set to 1, indicating that the address prefixes contained in the PIE's binding list are intended to be interpreted as IP version 4 (IPv4). If either the sender or the receiver of this PIE is using ATM switching hardware to implement the tag switch forwarding path, the Blist Type field is set to 6 ("32-bit downstream assigned VCI tag")

15   to indicate that, as will be seen below, the tag has a format and location specific to the ATM protocol. Otherwise it is set to 2, which means "32-bit downstream assigned." *Downstream assigned* means that a tag's meaning is being set by the router that will base its routing decisions on it, as opposed to the router that will tag the packet with it. The next, Blist Length field gives the length in bytes of the Binding-List field, and the op-

20   tional-parameters field is sometimes included to present related information.

Of these fields, the field of most interest here is the Binding-List field, whose format Fig. 3's fifth row depicts. That field contains one or more entries. When the Blist Type is 2, each of the entries includes precedence, tag, prefix-length, and prefix fields, as Fig. 3's fifth row indicates. To bind tag T to prefix X, the prefix-length field contains

25   X's length in bits, the prefix field contains X's value right padded with as many bits as needed to make it end on a byte boundary, and the precedence field is an eight-bit field that specifies the precedence with which the router that issued the PDU will service traffic that bears T as a tag.

So to request that a neighbor router use a given tag value when it forwards pack-

30   ets destined for a given prefix, a router sends a TDP message containing a

TDP_PIE_BIND type PIE whose binding-list portion's tag and prefix fields respectively contain that tag and prefix.

Now, PE1 uses this mechanism to ask that P1 bind to PE1's own address a distinguished tag value that means "pop the tag stack." (It makes a similar request to any other of the SP's transit routers to which it is directly connected.) The purpose of this request is to establish PE1, an edge router, as one that should see the lower, ultimate-destination-designating tag (T3 in Fig. 1) hidden from the transit routers. As a result of PE1's having advertised its host route, P1 already has an FIB entry that maps PE1's address to a next hop of PE1 and a tag-stack operation of "no op." As was stated above, the SP's routers are required to create TIB entries for all prefixes that they have FIB entries for, so P1 assigns a tag T1 to PE1 by creating a TIB entry that maps T1 to the destination PE1. And, in accordance with PE1's bind request, that entry's tag-stack operation is "pop the stack."

P1 must also distribute the new tag, so it uses TDP to ask that P2 use the T1 tag whenever it sends P1 a packet destined for PE1.

PE1's advertisement of its host route has resulted in P2's already having a FIB entry that maps PE1's address to a next hop of P1 and a tag-stack operation of "no op." P2 now modifies this FIB entry so that the tag-stack operation is "push T1."

Since PE1 is a destination in P2's FIB, P2 must bind a tag value to PE1's address. That is, it creates a TIB entry that maps T2 to a next hop of P1—i.e., to its FIB's next-hop entry for PE1—and to a tag-stack operation of "replace the top tag value with T1." P2 then uses TDP to ask that PE2 use tag value T2 whenever it sends P2 a packet destined for PE1.

PE2 already has a FIB entry that maps PE1's address to a next hop of P2 and a tag-stack operation of "no op." In response to P2's TDP message, PE2 now modifies this FIB entry so that the tag-stack operation is "push T2."

3. EBGP Messages from CE Routers to PE Routers

So far we have described only the tag binding that results from the routing information that the SP's routers have used an IGP to share with each other. But the present

invention is intended to be used to implement a peer-model VPN, so the client enterprise, too, shares routing information with some of the SP's routers.

The CE1 router is a routing adjacency of the PE1 router. That is, when CE1 forwards a packet destined for a remote system that can be reached through PE1, CE1 explicitly directs that packet to PE1. In the illustrated example as it will be elaborated on in connection with Fig. 2, it performs the explicit direction by encapsulating the packet in a link-level header containing PE1's hardware address on a common multinode network. In other configurations, it may do so by, for instance, placing that packet on a point-to-point link with or by sending the packet in transmission cells whose headers include a code that represents a channel between CE1 and PE1. Yet another way of providing the explicit direction is to use, e.g., encapsulated IP, whereby the packet includes an IP datagram whose destination address is PE1's network address but whose payload is another IP datagram, this one having the destination address of the remote destination. In this way, an internetwork route between CE1 and PE1 acts as a "link" in a higher-level internetwork route.

In contrast, CE1 is not in general a routing adjacency of CE2. That is, even when CE1 forwards a packet destined for a remote system reachable through CE2, it never explicitly specifies CE2 as a router through which the packet should pass on the way. True, the fact that CE2 is in the route may have been included in the reachability information that CE1 amassed in the course of filling its forwarding-information database. But in the course of actually forwarding a packet, CE1 simply notes that PE1 is the next hop to the ultimate destination.

In the Fig. 1 topology, suppose that CE1 is to tell PE1 which hosts are reachable at its site. For this purpose, it must use an external routing protocol, and we have assumed for the sake of example that it uses BGP. Together with RFC 1655, RFC 1654 and its predecessors describe that protocol's operation exhaustively, and we will not repeat that description here. For present purposes, we mention only a few features of most interest to the illustrated embodiment's operation.

As Fig. 4's first row indicates, BGP uses the TCP transport protocol. Concatenation of TCP-segment payloads results in a data stream in which the BGP application

looks for a predetermined marker sequence. It interprets the marker and subsequent fields as a BGP message header that contains information such as the message's length and type. To share routing information, the type of message that CE1 uses is the BGP "Update" message, whose format Fig. 4's second row depicts.

5   The drawing uses a section labeled "header+" to represent the header and a number of fields not of particular interest to the present discussion. The message ends with a list of interface address prefixes referred to as Network-Level Reachability Information (NLRI), and a Path Attributes field describes a path to hosts whose IP addresses begin with those prefixes. A Path Attribute Length field ("PAL" in the drawing) tells how long 10   the Path Attributes field.

In the present example, let us suppose that CE1 is at a site where all the hosts have IP addresses whose first byte is 10 ($0A_{16}$)and whose second byte is 1 (01_{10}). That is, they can be represented by the two-byte prefix $0A01_{16}$ (which the literature conventionally represents as "10.1.") To communicate this, CE1 places in an NLRI-field length 15   segment an indication that the prefix to follow is two bytes in length, and it puts $0A01_{16}$ in the following, prefix field, as Fig. 4's third row indicates.

Fig. 4's third row depicts the message's path-attributes portion as having three attribute fields, of which Fig. 4's fourth row illustrates one in detail. Attribute fields take the <type, length, value> form. The type field's second, "attribute code" half is shown as 20   containing the code value of 2, which indicates that the value field is to be interpreted as describing a path to the hosts that the message advertises as being reachable. Specifically, it is to be interpreted as listing the "autonomous systems" that have to be traversed to reach those hosts.

Now, whenever a system has a BGP connection of any sort, it must use an 25   Autonomous System Number (ASN). This is a number that the assigned number authority issues so that independently administered systems can identify each other when they use an external routing protocol. An "autonomous system" (AS) is a system under administration separate from others, and connection among an AS's hosts, whether direct or indirect, must be possible by way of the AS's resources only. Since CE1 cannot communicate with CE2 without using the SP's resources, the customer-enterprise-administered 30

28

resources comprise at least two ASs. So we will assume that CE1's ASN is A1, CE2's ASN is A2, and the PE routers' ASN is A3.

From PE1, only AS A1 is involved in reaching the hosts represented by prefix 10.1. To indicate this, the AS-path attribute's value includes a first field that identifies it a sequence of ASs, a second field that gives the number of ASNs in the list as one, and a third field that contains the list's sole ASN, A1.

Fig. 4's fifth row depicts another of that message's attribute fields, one whose attribute-code byte identifies it as specifying the "next hop" to be used in reaching the advertised host-address range. The value field contains CE1's address, thereby indicating that CE1 can forward traffic to those reachable destinations.

So CE1 has told PE1 that it undertakes to forward traffic to hosts whose IP address prefixes are 10.1. In response, PE1 assigns a tag, T3, to that address prefix in CE1's VPN, VPN V. (Actually, PE1 may use the same tag value for every address prefix mentioned by CE1.) In its TIB, PE1 creates an entry, indexed by this tag value, that specifies CE1 as the next hop. The entry specifies a tag-stack operation of "pop the tag stack" so that the tag used will be discarded to reveal the network-layer header to CE1.

4. IBGP Messages from PE1 to PE2

Additionally, PE1 sends BGP update messages to certain other of the SP's routers to tell them that they can forward to PE1 any packets destined for hosts whose addresses are in the 10.1 range. But PE1's SP network provides service to other customer enterprises that may also have 10.1-prefix hosts: those hosts' addresses may not be unique. So the SP assigns a different VPN identifier to each of its customers' VPNs. In the case of CE1's enterprise, let us assume that the code is a 16-bit identifier V. PE1 prepends the VPN identifier V to the IPv4 address prefix (10.1 in the example) and uses it in the BGP message to the other provider routers.

Indeed, the SP may assign VPN V more than one VPN identifier. A reason for doing so could arise if VPN V uses the SP not only as its backbone but also as its connection to outside systems, such as the SP's other customers or the public internet. In addition to the above-described reachability advertisement, which VPN V does not intend

the SP to share with systems outside the VPN, CE1 or another of VPN V's edge routers could also send PE1 information regarding routes over which VPN V would permit outside-origin traffic. For example, one route to a given node may be shorter and thus preferred for traffic from within the VPN, but a different route to the same node may include a firewall and therefore be preferred for traffic from outside the VPN. CE1 could specify the permitted scope of dissemination by using, say, the BGP communities attribute (RFC 1997) in the update message, or it could distinguish between different dissemination scopes by using separate channels between it and PE1 (e.g., by using different ones of PE1's IP addresses) for the different scopes.

PE1 must make this distinction in BGP messages that it sends to others of the SP's routers, because the roles of various SP routers as edge and transit routers is not in general the same for intra-VPN traffic as they are for inter-VPN traffic. To distinguish between different routes to the same destination, PE1 may prepend a first VPN identifier, say, V_I, to prefixes in routes intended only for intra-VPN advertisement and a second identifier, say, V_E, to prefixes in routes whose extra-VPN advertisement is permitted. Further identifiers may be used for further dissemination scopes. For the sake of discussion, though, we will assume that VPN V uses the SP as its internal backbone only and that the SP has accordingly assigned VPN V only one VPN identifier.

In accordance with the present invention, transit routers do not need the reachability information that CE1 has shared with PE1. So PE1 does not send the BGP message to the transit routers, and it may not send it to all edge routers. But Fig. 1 depicts only one other edge router, router PE2, and PE1 does send the BGP message to PE2, because that router is connected directly to VPN V. Those skilled in the art will recognize, though, that the message does not have to be sent as part of an actual BGP session between PE1 and PE2. In some large service providers, it is not considered practical for each BGP speaker to maintain BGP sessions with all other BGP speakers. So "route reflectors" act as intermediaries, maintaining sessions either directly or through other route reflectors with each of the BGP speakers and thereby propagating the necessary routing information. In that way, the number of IBGP sessions increases only linearly with the

number of BGP speakers. But the diagram shows only two PE routers, so it includes no route reflectors.

Regardless of how PE1 sends the message, Fig. 5 illustrates that message's format. Since it is a BGP update message, its format is similar to the one that CE1 sent to

5   PE1. Instead of using the conventional NLRI field to contain reachability information, though, PE1 obtains the greater format flexibility needed for the VPN-IPv4 address by using a "multiprotocol reachability information" type of attribute field, which has its own NLRI subfield. As Fig. 5's fourth row indicates, this type of attribute's code is 14. The first three octets of this type of field specify the address family that the attribute value

10 will use to represent the reachability information in the NLRI field, and Fig. 5's fifth row shows that PE1 assigns these bytes a value representing the Tagged VPN-IPv4 format. As Fig. 5's sixth row illustrates, the Tagged VPN-IPv4 format starts with a four-byte tag, whose value is T3 in the example. This is followed by a field representing the prefix-field length, which is four bytes in the example. The prefix field's first two bytes encode

15 the value V, which identifies the VPN, and the second two bytes have the value $0A01_{16}$, i.e., the sixteen-bit address prefix 10.1.

The other fields that Fig. 5's fifth row depicts include a next-hop field and a field that tells how long the next-hop field is. The next-hop field contains a six-byte VPN-IPv4 address whose first two bytes are zero—the next hop is not one of the customers'

20 routers—and whose remaining four bytes are PE1's IP address. the section below enti- tled "Internet Draft: Multiprotocol Extensions for BGP-4" describes messages of this general type in more detail.

In response to this message, PE2 extracts the NLRI field's VPN-IPv4 value and decodes it into a VPN identifier and an IPv4 address prefix. In its FIB for that VPN, it

25 creates an FIB entry that maps the IPv4 prefix to a next hop and a tag-stack operation. The next-hop value is PE1's address (since PE1's address appeared in the message's next-hop field). The tag-stack operation is "push tag value T3 onto the tag stack." Since PE1 is not a direct neighbor of PE2, this is a recursive FIB entry.

Note also that BGP Update messages concerning VPN-IPv4 address prefixes

30 cause modification only of the VPN-specific FIB, not of the general FIB. However, if the

original BGP message from CE1 had indicated that the reachability information could be disseminated beyond VPN V (or a broader dissemination scope could be inferred from, e.g., the channel by which it came, then PE2 would additionally install that IPv4 prefix, next hop, and tag-stack operation in the FIBs for all the VPNs to which that information's dissemination.

Although the illustrated embodiment employs only a single service provider to provide the VPN's backbone, there is no reason why more than one SP, whose facilities constitute more than one autonomous system, cannot cooperate to implement the present invention's teachings. In that case, the tag-binding and reachability information would further flow from one SP to the next by EBGP in the Fig. 5 format.

Specifically, the egress PE router in one of the SP networks could use BGP to distribute a tag binding for a particular VPN-IPv4 address to the BGP border router between the two SP networks. That BGP border router would then distribute a tag binding for that address to the ingress PE router.

5. EBGP Message from PE2 to CE2

PE2 then relays this information to CE2 by sending it an EBGP message similar to the one that CE1 sent to PE1. As Fig. 6 shows, this message's NLRI field indicates that hosts whose addresses begin with prefix 10.1 are reachable, its next-hop attribute field indicates that the next hop in the route to those hosts is PE2, and the AS-path attribute field indicates that the path to that prefix traverses autonomous systems A1 and A3.

When CE2 receives this message, it creates an FIB entry that maps prefix 10.1 to a next hop of PE2. Note that CE2 need not support tag switching. CE2 must also use its own IGP to inform other routers (not shown) at its site that it has a route to hosts whose addresses begin with prefix 10.1.

6. Tracing a Data Packet

As a result of these operations, the various routers have the routing information that they need when CE2 sends to PE2 a data packet P whose destination address is 10.1.0.1, which Fig. 1 depicts as "D1"

To send P, CE2 looks up address 10.1.0.1 in its FIB and finds that the longest matching address prefix is the sixteen-bit prefix 10.1. The corresponding next hop is PE2. CE2 is directly attached to PE2, so it forwards P to PE2 over the data link connecting the two routers.

5   PE2 receives packet P and notes that it received that packet from a particular VPN, VPN V. For the sake of simplicity, we assume that PE2 concludes this from the fact that it receives the packet over a point-to-point interface dedicated to communication with CE2. But edge routers can base that determination on other factors instead. For example, suppose that the interface the interface is a local-area-network interface over

10   which packets from different VPNs could arrive. In that case, CE2 might rely on the data-link source address and base the determination on its knowledge of the VPN's constituent systems. Other implementations may base the source determination on cryptographic authentication data that the packet contains. In a similar vein, the log-in procedure performed by a customer contacting the PE router by way of a dial-in link may re-

15   sult in the PE router's obtaining information from an authentication server, and it may base its identification of the source VPN on this further information.

In any event, the PE router identifies the source VPN, and the source VPN in this case is VPN V. So PE2 looks up P's destination address in its FIB that is specific to VPN V. It finds that the longest matching address prefix is the sixteen-bit prefix 10.1. (In

20   this example, which focuses on intra-VPN communication, we assume that PE2 further infers from the source determination that the packet is not to be permitted outside VPN V, so PE2 would not look further if it failed to find a match. In other circumstances, though, PE2 might look in the FIBs of other VPNs, which may have indicated their availability to forward packets to that address.) The corresponding next hop is PE1,

25   and the tag-stack operation is "push T3 on the tag stack." So PE2 creates a tag stack for P and pushes T3 onto it. Since PE2 is not directly connected to PE1, P2 performs a recursive lookup in its general FIB.

We know from the preceding discussion that PE2 has an FIB entry corresponding to PE1's thirty-two-bit address, that the next hop in that FIB entry is P2, and that the tag-

30   stack operation in that FIB entry is "push T2 onto the tag stack." So PE2 pushes T2 onto

P's tag stack. The stack now has two tags; the top tag T2, and the bottom tag is T3. PE2 tags P with this stack and sends P over the data link to P2, as Fig. 1 shows diagrammatically.

When P2 receives packet P, it attempts to forward it by looking up T2 in its TIB.
From the tag-distribution discussion, we know that T2 maps to a TIB entry whose next hop is P1 and whose tag-stack operation is "replace the top tag value with T1." So P2 performs the tag-stack operation and sends the packet over the data link to P1. (At this point, packet P's top tag is T1, and its bottom tag is T3.)

When P1 receives packet P, it attempts to forward it by looking up T1 in its TIB.
We know from the tag-distribution discussion that T1 maps to a TIB entry whose next hop is PE1 and whose tag-stack operation is "pop the tag stack." So P1 performs the tag-stack operation and sends the packet over the data link to PE1. (At this point, packet P is carrying only one tag, T3.)

When PE1 receives packet P, it attempts to forward it by looking T3 up (which is now at the top of the stack) in its TIB. We know from the tag-distribution discussion that T3 maps to a TIB entry whose next hop is CE1 and whose tag-stack operation is "pop the tag stack." So PE1 performs the tag-stack operation and sends the packet over the data link to CE1. Note that PE1 has popped the last tag off the tag stack before sending the packet to CE1. So CE1 receives an untagged packet, which it forwards in the conventional way.

Now, although we introduced the foregoing example with Fig. 2's illustration of Ethernet as the link-level protocol, those skilled in the art will recognize that other protocols can readily be substituted. The adaptations required for that purpose are largely straightforward and do not in general require separate discussion. But there may be some value in briefly discussing an Asynchronous Transfer Mode (ATM) example, because such an adaptation moves part of the tag stack to the ATM header.

To that end, we consider Fig. 7, whose topology is identical to that of Fig. 1, but we assume that P1 and P2 are ATM switches and that PE1 and PE2 are routers that attach to P1 and P2, respectively, over ATM interfaces. Fig. 8 depicts the typical data message

that, say, PE2 would send to P2 in such an arrangement. Fig. 8 is best understood by comparison with the second row of Fig. 2's Ethernet example. In that diagram, the Ethernet header (DEST. ADDRESS, SOURCE ADDRESS, and TYPE) and trailer (CRC) encapsulate a payload in the form of tag fields and an IP datagram. Fig. 8's third row depicts an ATM frame, and that drawing's fourth and fifth rows show that the frame's payload is similar to that of Fig. 2's Ethernet frame. The only difference in the payloads is that Fig. 8's fifth row represents the left (top) tag by question marks, which indicate that the top tag's contents do not matter.

The reason why they do not is that the routing decisions made by Fig. 1's P2 on the basis of those contents are made by Fig. 7's (ATM) router P2 on the basis of an ATM VPI/VCI field in the header of an ATM "cell." From the point of view of an ATM client, the frame of Fig. 8's third row is the basic unit of transmission, and it can vary in length to as much as 64 Kbytes of payload. (Those skilled in the art will recognize that there are also other possible ATM frame formats, but Fig. 8's third row depicts one, known as "AAL5," that would typically be employed for user data.) For communication between ATM switches, however, ATM actually breaks such frames into fixed-size cells.

Each cell consists of a header and a payload, as Fig. 8's second row illustrates. Among the purposes of the header's PTI field, depicted in Fig. 8's first row, is to indicate whether the cell is the last one in a frame. If it is, its last eight bytes form the frame trailer field that Fig. 8's third row depicts. Among other things, the trailer indicates how much of the preceding cell contents are actual payload, as opposed to padding used to complete fixed-size cell.

The only other header field of interest to the present discussion is the VPI/VCI field of Fig. 8's first row. As is well known to those skilled in the art, ATM systems organize their routes into "virtual channels," which may from time to time be grouped into "virtual paths." Each switch associates a local virtual path/virtual channel indicator (VPI/VCI) with a channel or path that runs through it. When an ATM switch receives a cell, it consults the cell's VPI/VCI field to identify by table lookup the interface by which to forward it, replaces that field's contents with a value indicated by the table as being the next switch's code for that path or channel, and sends the resultant cell to the next switch.

In other words, the function performed by the VPI/VCI field enables it to serve as the stack's top tag.

So PE1 will bind a VPI/VCI tag, call it VC1, to the address of PE1 and distribute that binding to P1. P1 will bind a VPI/VCI tag, call it VC2, to the address of PE1 and distribute that binding to P2. P2 will bind a VPI/VCI tag, call it VC3, to the address of PE1 and distribute that binding to PE2.

Now, when PE2 receives from CE2 a packet destined for a site that is in CE2's VPN and is reachable via CE1, it does the following.

First, it looks up the destination address of that packet in its VPN-specific forwarding table. It finds a recursive entry whose tag operation is "push on T3". On performing the recursive lookup, it finds that the next hop is an ATM switch and that the tag value is the VPI/VCI value VC3. It accordingly forms the frame depicted in Fig. 8's bottom three rows. It then breaks the frame into cells of the type that Fig. 8's top two rows depict, placing the VC3 value in the VPI/VCI field, and sends them in sequence to P2.

P2, on a cell-by-cell basis, replaces VC3 with VC2 and forwards the resultant cells to P1. Similarly, P1 replaces VC2 with VC1 on a cell-by-cell basis and forwards the resultant cells to PE1. PE1 eventually collects all the frame's cells and reassembles them. PE1 then extracts the resultant frame's user data, pops the tag stack, and forwards the resultant frame in accordance with the resultant tag stack (which now contains a single tag, T3). Note that in this scenario it is PE1, not P1, that pops the stack to get to the tag, T3, that indicates the extra-SP route. This is because P1 in this scenario is an ATM switch, and ATM switches do not have the capability of popping the stack themselves.

In the foregoing ATM example, the top tag in the tag-stack field never has any meaning. But now suppose that only P1 is an ATM switch: P2 and PE1 are routers attached to P1 via ATM interfaces. Then the PE2-P2 link would contain Figure 2-style packets, P2 would base its decision on the top tag-field tag, and it would forward ATM cells in response.

Considerations for Extension to Inter-VPN Use

To avoid obscuring the present invention's basic principles, the foregoing discussion focused mainly on their use for intra-VPN communication. We now turn to suggestions for ways in which systems that employ those teachings could perform inter-VPN communication.

1. INTERNAL VS. EXTERNAL VPN-IPV4 ADDRESSES

As was explained above, it may be necessary to maintain two routes to a particular IPv4 address exported from one VPN to another. One route is used for intra-VPN traffic, and the other is used for inter-VPN traffic. When a particular IPv4 address is exported from one VPN to another,. For example, suppose that the system bearing a particular address is in site S1. Intra-VPN traffic to that system should certainly go directly to S1. However, there may be a firewall located at site S2, and it may be desired to pass all inter-VPN traffic through that firewall. In this case, inter-VPN traffic to the system in question should travel via S2.

In order to be sure that BGP can simultaneously install an intra-VPN and an inter-VPN route to the same address, it is necessary to use a different VPN-IPv4 address for intra-VPN connectivity than for inter-VPN connectivity.

Therefore, each VPN will have two VPN IDs. One will be the "Internal VPN ID," and one will be the "External VPN ID."

Each PE router will translate the IPv4 addresses from its attached VPNs to one or the other or to both of these VPN-IPv4 addresses. The rules for doing so will be discussed later.

A VPN-IPv4 address whose VPN ID is the Internal VPN ID of its VPN must not be distributed by any PE router to any CE router, unless that CE router is in that VPN. To prevent any unintended redistribution, a PE router that distributes an IPv4 address to another PE router must assign it the NO_EXPORT Community Attribute. According to RFC 1997, "BGP Communities Attribute," this attribute means:

> All routes received carrying a communities attribute containing this value MUST NOT be advertised outside a BGP

confederation boundary (a stand-alone autonomous system that is not part of a confederation should be considered a confederation itself).

As we shall see below, this will prevent the corresponding address from being advertised outside the VPN. (One could instead define a new Community Attribute value, e.g., NO_EXPORT_OUTSIDE_VPN, for this purpose, but NO_EXPORT seems adequate and makes it easy to accommodate the case where the CE router itself specifies a NO_EXPORT attribute.

(An alternative would be to install filters that prevent VPN-IPv4 addresses with Internal VPN IDs from being transmitted outside a BGP confederation. This could be done if one could tell by inspection that a particular VPN ID is Internal, rather than External.)

2. AUTONOMOUS SYSTEM NUMBERS

a. ASN USED BY PE ROUTERS ON IBGP CONNECTIONS

Since the PE routers (in the same P-network) are to use IBGP to distribute routes among themselves, it follows that there must be some Autonomous System Number (ASN), known to all the PE routers, which they use when setting up these connections. (A BGP connection is not treated as an IBGP connection unless both BGP speakers have used the same ASN.)

If the P-network is already in use as an internet transit network, it will likely already have a globally unique ASN, and this can be used on these IBGP connections.

b. ASN USED BY CE ROUTERS ON EBGP CONNECTIONS

When a particular site is a "stub site," it is not necessary for the CE router to talk BGP to the PE router, though under certain circumstances it may be desirable for it to do so. However, whenever a particular site has a C router that is talking BGP to another C router, then the CE router will need to talk BGP to the PE router. This is true whether the C routers talking BGP are talking to other C routers at the site, to other C routers at different sites of the same VPN, to other C routers of different VPNs, or even to routers in the public internet.

When a CE router distributes routing information to a PE router, the intention is that the information ultimately be distributed to one or more other CE routers. One PE router uses IBGP to distribute the information to another, and the latter redistributes it to another CE router.

5   Since routes learned over IBGP are in general not redistributed over IBGP, and since PE routers have IBGP connections to each other, it follows that the CE routers must talk EBGP to the PE routers. Each site where a CE router talks EBGP to a PE router must have an ASN. Call this a "Site ASN."

The number of globally unique ASNs is limited, and it is not feasible to assign 10   one to each individual VPN site. There is however a "private ASN" numbering space containing 1023 ASNs, which a service provider can administer as he sees fit. So the Site ASNs must be taken from the private ASN space. Since the size of the private ASN space is limited, it is desirable to use the same ASN numbers in different VPNs.

This can be done by modeling each VPN as a "BGP Confederation." This means 15   that the CE router and the PE router do not run "regular" EBGP between them: they run "Confederation EBGP (CEBGP)." CEBGP uses some of the procedures of regular EBGP, some of the procedures of IBGP, and some procedures of its own. However, these procedures are all well-defined and implemented.

3. USING BGP-CONFEDERATION TECHNIQUES FOR AS-PATH
20   MANIPULATION

A BGP confederation is a set of Autonomous Systems (ASs) that appear as a single AS to all ASs not in the Confederation. Only within the Confederation are the component ASs visible. That is, externally to the Confederation, the Confederation has a single ASN. Within the Confederation, each "Member AS" of the Confederation has its 25   own ASN, which is distinct from the Confederation's ASN. The distinction shows up primarily in BGP Confederation procedures for AS-path manipulation, which we recommend for inter-VPN communication. (This does not imply that BGP Confederation procedures affecting other attributes should also be used.)

BGP maintains loop freedom by associating an AS-path with each route. Roughly, this is a list of the ASs through which a packet must travel to reach the destination. When a router distributes a route via EBGP, it adds its own ASN to the AS-path. When a router receives a route via EBGP, it checks to see if its own ASN is already in the AS-path. If so, it discards the route, in order to prevent the loop.

With Confederations, this procedure is slightly changed. When a router distributes a route on a CEBGP connection, it adds its own AS to the AS-path, but it marks that AS as being within the Confederation. When a router that is within a Confederation distributes a route on an EBGP connection, it first removes from the AS-path all ASs that are marked as being within the Confederation. Then it adds the Confederation's ASN to the AS-path.

When a router that is in a Confederation receives a route over an EBGP connection, it will discard the route if the AS-path contains the Confederation's ASN. When a router receives a route over a CEBGP connection, it will discard that route if the AS-path contains the Member ASN of that router, and that Member ASN is marked as being within the Confederation.

Since the Member ASNs of a Confederation are never seen outside the Confederation, they can be assigned from the Private ASN space.

In a VPN, each site containing a CE router that talks BGP to a PE router would have a Site ASN taken from the Private ASN space. Then these Site ASNs need be unique only within a single VPN: they can be reused in other VPNs. The P network is part of each such Confederation and needs to have a Member ASN that can be used within each Confederation. The P network can have a single ASN that it uses as its Member ASN in all Confederations. If it has a globally unique ASN, this can be used.

If a VPN spans multiple service providers, then its Site ASNs must be unique across all the providers, and each P network must use a globally unique ASN.

When a router receives a route whose AS-path contains its site number, it conventionally rejects the route if the site number is not marked as being part of the confederation, and it is preferable for CE routers to follow this policy. Otherwise, since a VPN

is modeled as a Confederation, care must be taken to ensure that whenever two C routers in the same VPN have a direct BGP connection with each other (i.e., a "backdoor" connection between routers in the same VPN, at the same or different sites), they talk either IBGP or CEBGP, never regular EBGP. When talking CEBGP, each router would use its Site ASN as its ASN, for the purpose of (a) filling in the "My Autonomous System Number" field in the BGP Open message, and (b) adding its ASN to the AS-path.

When a PE router receives from a CE router over a CEBGP connection, routes to IPv4 addresses, the PE router will immediately translate those addresses to VPN-IPv4 addresses, using the Internal VPN ID of the CE's VPN. ("Immediate translation" means that the addresses appear in BGP's "adj-rib-in" table as VPN-IPv4 addresses.) When a PE router distributes VPN-IPv4 addresses to a CE router over a CEBGP connection, it first converts them to IPv4 addresses by stripping off the VPN ID.

The External VPN ID of a particular VPN can have the same value as the VPN ASN. The Internal VPN ID must have a different value. It may be convenient for these values to be algorithmically related, but this is not required.

If a VPN spans multiple service providers, its Internal VPN ID and its External VPN ID must be globally unique. Otherwise, they must be unique only within the scope of a single service provider. Note also that any quantity that is used as an External VPN ID of one VPN may not be used as an Internal VPN ID of any other VPN, and vice versa.

4. INTER-VPN COMMUNICATION AS COMMUNICATION BETWEEN TWO CONFEDERATIONS

Since each VPN is modeled as a BGP Confederation, each VPN appears as an AS to each other VPN. Communication between two VPNs is modeled as communication between two ASs, using the P network as the transit AS. Therefore if a CE router uses BGP to export routes, via a PE router, to another VPN, it must do so via a regular EBGP connection to the PE router. Of course, on the EBGP connection it uses the VPN ASN, not the Site ASN.

If the P-network has a globally unique ASN, it can be used both within a Confederation and between Confederations.

Whenever two C routers in different VPNs have a direct BGP connection with each other (i.e., a "backdoor" connection between routers in different VPNs), care must be taken to ensure that they talk EBGP with each other. When talking (non-confederation) EBGP, each router would use its Confederation ASN as its ASN for the purposes of (a) filling in the "My Autonomous System Number" field in the BGP Open message, and (b) adding its ASN to the AS-path.

So in the most general case, a CE router may need to have two BGP connections to a PE router, an EBGP connection (for inter-VPN connectivity) and a CEBGP connection (for intra-VPN connectivity). There may be only one BGP connection between a given pair of IP addresses. So if a given pair of routers need to have two BGP connections between them, each router must use a distinct address on each connection.

When a PE router receives, from a CE router over an EBGP connection, routes to IPv4 addresses, the PE router will immediately translate those addresses to VPN-IPv4 addresses, using the External VPN ID of the CE's VPN. When a PE router distributes VPN-IPv4 addresses to a CE router over an EBGP connection, it will first convert them to IPv4 addresses by stripping off the VPN ID.

A site in a VPN may maintain a backdoor connection to the public internet, via an EBGP connection. If this EBGP connection is not via the same service provider that is providing the VPN, the VPN ASN must be from the public AS numbering space. Otherwise, it may be from the private AS numbering space, and the C router maintaining the EBGP connection to the internet should be configured to strip all private ASNs from the AS-path.

In general, P routers with EBGP connections to routers outside the P network will not accept routes to VPN-IPv4 addresses over those connections. To do so would allow routers outside the Service Provider's control to spoof routes to the VPN, thereby compromising the security that the customer expects. If it is necessary to make any exceptions to this rule (to support, say, multi-provider VPNs), the security effects of those exceptions would need to be carefully considered.

5. HOW TO DETERMINE WHEN A CE ROUTER NEEDS ZERO, ONE, OR TWO BGP CONNECTIONS TO A PE ROUTER

If the CE router's site does not have any backdoor connections, neither a CEBGP nor an EBGP connection is necessary. In this case, all the information that would be passed via BGP can be statically configured in the PE router. The site will not have a Site ASN. IBGP between PE routers is still used to pass routing information about one site to the others.

By a "backdoor connection," we mean a BGP connection between a C router at the site and any router other than a PE router. If two sites in a particular VPN are inter-connected via static routing and/or IGP, then we model them as a single site, rather than as two sites with a backdoor connection.

Even in the absence of backdoor connections, it can be desirable to use BGP be-tween the CE and the PE router, if the site has a significant number of address prefixes that are sometimes up and sometimes down, or if there are address prefixes that move from one site to another. This can also be desirable simply as a way to avoid the configu-ration task associated with static routing.

If the CE router's site does not have any backdoor connections to other VPNs (or to the public internet), but it is desired to have a BGP connection to the PE router (either for the reason given in the prior paragraph, or because there are backdoor connections to other sites in the VPN), it is necessary to have a CEBGP connection between CE router and PE router. As we will see below, routes distributed over CEBGP will not thereby be distributed to any other VPN. However, distribution of routes to other VPNs can still be achieved via configuration of the PE router.

If the CE router's site has backdoor connections to other VPNs (or to the public internet), and if it serves as a transit network for traffic from other VPNs (or the public internet), then the CE router must run EBGP with the PE router, in order to properly dis-tribute the routes for which it is a transit network.

If a VPN has multiple sites that have EBGP connections to PE routers, then there must also be a CEBGP connection from each of those sites to a PE router.

6. USING COMMUNITY ATTRIBUTES TO CONTROL THE EXPORTING OF ADDRESSES FROM ONE VPN TO ANOTHER

As stated previously, whenever a PE router uses IBGP to distribute to another PE router (or route reflector) a route to a VPN-IPv4 address, the NO_EXPORT Community

5   Attribute will be included as an attribute of that route if the VPN ID of that address is an Internal VPN ID.

When a PE router uses IBGP to distribute a route to a VPN-IPv4 address to another PE router (or route reflector), and the VPN ID of that address is an External VPN ID, Community Attributes must be included that specify the set of VPNs to which the

10   address in question is to be exported.

This requires a distinguished class of Community Attributes that are used only for this purpose. In general, when such attributes are received by P routers over EBGP connections, they should be removed (via inbound filtering), unless there is explicit configuration of the P router that allows them to be passed on unchanged.

15   The Community Attribute that is used to indicate that an address is to be exported to a particular VPN should be algorithmically derivable from that VPN's ASN, and vice versa.

If a CE router talks EBGP to a PE router, the CE router may, with each address it distributes, include a set of Community Attributes, indicating the set of other VPNs (pos-

20   sibly including the public internet) to which the address is to be exported. If so, the PE router may be configured with a set of addresses from the C network that the CE router is authorized to export to a set of other VPNs. In that case, the PE router will remove (via inbound filtering) any unauthorized Community Attributes sent by the CE router. The PE router may be configured with a set of addresses from the C network that are to be ex-

25   ported to a set of other VPNs, even if the CE router does not include the necessary Community Attributes. In this case, the PE router must add (via inbound filtering) the missing Community Attributes.

When a PE router receives a route to an external VPN-IPv4 address and that route is associated with a Community Attribute that identifies the VPN of a CE router to which

30   that PE router is attached, then the route is a candidate for redistribution to the CE router.

44

(Of course, a VPN-IPv4 address is translated into an IPv4 address, by having its VPN ID stripped off, before being distributed to a CE router.)

The PE router may be configured to allow only particular VPN-IPv4 addresses to be distributed to a particular CE router, regardless of the Community Attribute. Or it may be configured to prevent the distribution of particular VPN-IPv4 addresses to a particular CE router, regardless of the Community Attribute. In such cases, outbound filtering should be used to prevent distribution of such addresses to the CE router.

7. A SLIGHTLY DIFFERENT WAY TO USE COMMUNITY ATTRIBUTES: CLOSED USER GROUPS

The Community Attribute can be used in a similar though somewhat different way to represent "Closed User Groups" (CUGs) of VPNs, rather than target VPNs.

A CUG is a set of VPNs. A CE router could associate a route to a particular address with one or more CUGs. The PE router would strip any CUGs that the CE router is not authorized to use. The PE router could also add additional CUGs, or could add CUGs when the CE router has not specified any. The PE router would need to know which VPNs are members of which CUGs, so it could determine which other PE routers it needs to distribute the routes to.

When a route with a CUG is received, it will be distributed over an EBGP connection to a CE router only if the PE router is configured with the knowledge that the CE router is a member of that CUG.

The use of CUGs may simplify the configuration of the PE routers.

8. IBGP BETWEEN PE ROUTERS

In conventional uses of BGP, the set of EBGP/CEBGP speakers in a given AS is supposed to be "fully meshed" (or "fully reflected" through route reflectors). Otherwise, there is no way to ensure that communication between any two points is possible. For VPNs, we do not want to require that communication between every pair of points be possible, so the PE routers need not in general be fully meshed. A PE router A needs to talk IBGP to a PE router B only if A and B both attach to CE routers in the same VPN, or

if A attaches to a CE router in VPN 1 that is exporting addresses to a VPN 2, and B attaches to a CE router in VPN 2.

For each PE router that is to be an IBGP peer of a given PE router, the given PE router will know which VPNs the peer is interested in. If a PE router A has an IBGP peer B, and B is interested in VPN 1, then A shall distribute a route to B if and only if one of the following two conditions holds:

—the address corresponding to the route is a VPN-IPv4 address in VPN 1, or

—one of the following conditions holds:

* the VPN ID of the VPN-IPv4 address is the Internal VPN ID for VPN 1, or

* the VPN ID of the VPN-IPv4 address is the External VPN ID for VPN 1, and the route has a Community Attribute that indicates that it should be distributed into VPN 1.

Each PE router, before distributing a route, will also assign a tag for that route. This will be encoded, in a way to be defined, as an attribute of that route.

When a PE router redistributes over IBGP a route received from a CE router (whether it is received over EBGP or CEBGP), it should always put itself in as the next hop. This ensures that the next hop is always reachable in the P network's IGP (i.e., it does not require routes to all the CE routers to be injected into the P-networks' IGP). It also ensures proper interpretation of the tag that the PE router assigns to the distributed address prefix; the tag associated with an address prefix should be a tag assigned by the "next hop" for that prefix.

For the purpose of supporting VPNs, PE routers need to support the following capabilities:

—Tag distribution via BGP

—VPN-IPv4 Address Family

—VPN "edge capabilities," i.e., whatever special procedures are needed in order to interact with the CE routers—e.g., translation between VPN-IPv4 and IPv4 addresses, per-VPN lookup tables, etc.

BGP Capability Negotiation, as described in the section below entitled "Internet Draft: Capabilities Neogation", should be used to determine whether an IBGP peer has the appropriate capabilities.

9. IBGP BETWEEN A PE ROUTER AND A P ROUTER THAT IS NOT A PE ROUTER

PE routers may have "ordinary" EBGP and IBGP connections that have nothing to do with VPNs. On such ordinary connections, IPv4 NLRI rather than VPN-IPv4 NLRI is used; routes learned from CE routers will not be sent on such connections, unless the PE router is configured to export those routes to the public internet.

Any router with a BGP connection to the internet must ensure, through proper filtering, that it does not leak any routes to the internet that are not part of the P network's AS, or of the AS of some client network of the P network. When routes are leaked to the internet, all private AS numbers must be removed (via outbound filtering) from the AS-path.

10. CONFIGURATION OF THE PE ROUTERS

Each PE router must be configured with the following information:

a. Per CE Router that attaches to the PE router

i. The address of the CE router to use when participating in a CEBGP connection.

The PE router may maintain a static route to this address and need not redistribute this address into the IGP of the P network (as long as the PE router always sets itself as the next hop before redistributing routes received from the CE router). In this case, the same address may be reused for other CE routers, subject to the constraint that all the CE routers attaching to a given PE router have distinct addresses. If the PE router distributes this address into the P network's IGP, though, the address should be a unique address in the P network's address space.

This parameter can be omitted if no CEBGP connection is to be formed.

ii. The address of the CE router to use when participating with it in an EBGP connection.

This parameter can be omitted if no EBGP connection is to be formed.

iii. The address of the PE router to use when participating in a CEBGP connection with the above-specified CE router.

iv. The address of the PE router to use when participating in an EBGP connection with
5 the above-specified CE router.

(Can be omitted if no EBGP connection is to be formed.)

v. The CE router's Site ASN.

This parameter can be omitted if no CEBGP connection is to be formed.

vi. The CE router's Internal VPN ID.

10 vii. The CE router's External VPN ID

This doubles as its VPN ASN if an EBGP connection is to be formed.

viii. A list of VPNs or CUGs to which the CE router can export addresses, and, for each such VPN, the set of addresses that are authorized to be exported to it.

The set of addresses may be "all." For each such set of addresses, there needs to
15 be an indication as to whether the PE router should allow the addresses to be exported if

the CE router attempts to export them, or whether the PE router should initiate the export

of the addresses independently of any action on the part of the CE router. (The latter

would be the only way to get export if there is no EBGP connection to the CE router.)

ix. A list of VPNs or CUGs that can export addresses to the VPN of the CE router, and,
20 for each such VPN, a set of addresses that are authorized for export into the VPN of the
CE router.

This set may be "all." For distribution of an address between the public internet

and a VPN, the public internet shall be represented as VPN 0.

b. Per VPN or CUG, for each VPN to which the PE router attaches via a CE router,
25 *and for each VPN or CUG to which one of the attached VPNs can export addresses:*
the set of PE routers interested in that VPN or CUG.

IBGP connections will be opened to all such PE routers. If these are provided by

only a few route reflectors, manual configuration is acceptable, but auto-discovery will be

required as a practical matter if they are provided by a large number of other PE routers.

If the PE router has a CEBGP connection to the CE router, the addresses to be distributed intra-VPN will be those addresses distributed by the CE router over the CEBGP connection. Otherwise, the PE router needs to be configured with those addresses, or it needs to obtain them in some other way (such as ODR or RIP).

If the PE router has an EBGP connection to the CE router, the addresses to be distributed inter-VPN will be those addresses distributed by the CE router over the EBGP connection. Otherwise, the PE router needs to be configured with those addresses.

11. CONFIGURATION OF THE CE ROUTERS

If the CE router is talking BGP to a PE router, the CE router will need to be configured to set up a CEBGP connection, or both a CEBGP and an EBGP connection, to a PE router. It must then be configured with an address of the PE router for each such connection. This address will be from the address space of the P network.

The CE router should have a static route to the PE router address. This route need not be redistributed into the C-network's IGP (though it should be safe to do so, because we are not trying to handle the case where there is addressing conflict between the C network and the P network).

The CE router does not use VPN-IPv4 addressing, and does not assign tags to the addresses it distributes to the PE router.

If the CE router is at a stub site, then:

—if it uses the same PE router(s) for intra-VPN as for inter-VPN traffic, it should be configured to have a default route pointing to the PE router(s), and should inject "default" into its IGP.

—if it uses a different PE router for inter-VPN traffic than for intra-VPN traffic, then it must be configured with appropriate static routes, and must inject them into its IGP.

(Even if the CE router talks BGP to the PE router, there is no reason to redistribute the BGP routes into the IGP.)

If the CE router is not at a stub site, then proper administration must be done to ensure that BGP routes and/or default routes are injected into the IGP in a proper manner.

12. DISTRIBUTION OF ROUTES FROM CE ROUTERS TO PE ROUTERS ON CEBGP CONNECTIONS

5 *a. CE router procedures*

A CE router will distribute all routes to all destinations on its site over its CEBGP connection to a PE router. Routes to destinations on other sites (through backdoor routes) may also be distributed to the PE router on the CEBGP connection; this is a matter of policy of the C network.

10 *b. PE router procedures*

When a PE router receives routes on the CEBGP connection, it will of course translate the IPv4 addresses to VPN-IPv4 addresses. It will also remove from each route any VPN Community attributes that may be present. It will add the NO_EXPORT community attribute, to prevent the route from being distributed out of the Confederation.

15 The PE router should check the AS-path of each route it receives from the CE outer to ensure that the appropriate Site ASN appears at the beginning.

13. DISTRIBUTION OF ROUTES FROM CE ROUTERS TO PE ROUTERS ON EBGP CONNECTIONS

a. CE router procedures

20 A CE router may distribute any routes to a PE router on an EBGP connection. However, it should avoid distributing any route on such a connection unless it intends to export that route to another VPN, or to the public internet.

b. PE router procedures

The PE router will ignore routes to any destinations that, according to the PE
25 router's configuration, are not to be exported to other VPNs (including the public internet).

If a route from the CE router does not have a Community Attribute associated with it, the PE router will, before further distributing it, add the VPN community for each other VPN to which the route may be exported, according to the PE router's configuration.

5   If a route from the CE router does have one or more Community Attributes associated with it, the PE router will remove any Community Attributes that do not correspond to VPNs to which the route may be exported, according to the PE router's configuration.

If the PE router allows a particular route to be exported to a number of VPNs, this
10   procedure allows the CE router to specify a subset of those VPNs to which it should be exported. If this is allowed, then the PE router must be able to detect when an EBGP update removes a Community Attribute that used to be there, so the route can be withdrawn from the corresponding VPN.

The PE router should check the AS-path of each route it receives from the CE
15   router to ensure that the correct value of the VPN ASN appears at the beginning. If not, the PE router may replace it with the correct value.

The PE router will convert all IPv4 addresses from the CE router to VPN-IPv4 addresses, using the External VPN ID of the CE router's VPN, before redistributing them. There is one exception: if a route is to be distributed to VPN 0, it should be dis-
20   tributed as an IPv4 address, without any Community Attribute. (This allows for distribution to the public internet via a BGP speaker that is not VPN-aware.)

14. DISTRIBUTION OF ROUTES FROM PE ROUTERS TO CE ROUTERS ON CEBGP CONNECTIONS

A PE router will distribute to a CE router, over a CEBGP connection, routes to all
25   VPN-IPv4 addresses whose VPN ID is the Internal VPN ID of the CE router's VPN. No other routes shall be distributed on this connection. The VPN-IPv4 addresses will be translated to IPv4 addresses before distribution.

The AS-path should be modified by prepending the P network's ASN.

15. DISTRIBUTION OF ROUTES FROM PE ROUTERS TO CE ROUTERS ON EBGP CONNECTIONS

A PE router will distribute a route with VPN-IPv4 NLRI to a CE router on an EBGP connection only if both the following conditions hold:

5 —the PE router is configured to allow the particular VPN-IPv4 address to be exported to the CE router, and

—the PE router received the router with a Community Attribute that corresponds to the VPN of the CE router, or to a CUG that is associated with that CE router.

10 This ensures that the route came from a proper place, and is going to a proper place.

Community Attributes that represent target VPNs or CUGs should be stripped before the route is distributed to the CE router.

VPN-IPv4 addresses should be translated into IPv4 addresses.

15 The AS-path should be modified by prepending the P network's ASN.

A PE router will distribute a route with IPv4 NLRI to a CE router on an EBGP connection only if the PE router is explicitly configured to allow that address to be exported to the CE router's VPN. This allows the VPN to import addresses from the public internet.

20 *Inter-VPN-Routing Example*

To illustrate the use of internal and external VPN IDs, Fig. 9 depicts a service-provider network simply as an oval, omitting all individual routers except PE1, PE2, and PE3. PE1 and PE2 are edge routers with respect to customer nodes in a first VPN, VPN A, and PE3 is an edge router with respect to a second VPN, VPN B. A target destination D in one VPN A is reached most directly through a customer edge router CE1 at

25 the same site. But VPN A has a firewall in CE2, and the policy is that any packets from outside VPN A must go through CE2 before they go to any VPN A destination.

In this situation, CE1 uses EBGP to advertise to PE1 its access to D. In some manner determined by local configuration, PE1 recognizes that advertisement as being only for VPN A consumption. For example, PE1 may be configured to recognize the interface used by CE1 as one that advertises only intra-VPN reachability, or CE1 may employ a NO_EXPORT value of the BGP community attribute in its advertisement. In any case, PE1 reports itself by IBGP as the next hop to destination Int(D) (where "Int(D)" represents the concatenation of VPN A's internal VPN ID with D's network address or prefix). Preferably, it knows which routers are edge routers with respect to VPN A and makes this advertisement only to them. Alternatively, it is not so discriminating, but it is only such routers that adjust their FIBs in accordance with that information.

In either case, PE2 thereby learns this information and uses it to construct an FIB entry in its per-VPN FIB corresponding to VPN A. (If, as the drawing does not show, PE3 attaches to a CE router that is in VPN A, then it, too, uses that information to construct an FIB entry in its per-VPN FIB corresponding to VPN A.)

Since CE2 is to operate as the firewall, it must advertise itself as according access to all systems that the enterprise is willing to accord extra-VPN visibility, so it also uses EBGP to advertise node D's reachability. In some manner determined by local configuration, PE2 recognizes that advertisement as being for extra-VPN A consumption, and it reports itself as the next hop to destination Ext(D) (where "Ext(D)" represents the concatenation of VPN A's external VPN ID with D's network address).

PE3 thereby learns this information and uses it to construct an FIB entry in its per-VPN FIB corresponding to VPN A. (If, as the drawing does not show, PE2 attaches to a CE router that is in VPN B, then it, too, would use that information to construct an FIB entry in its per-VPN FIB corresponding to VPN B.)

Now, when a packet addressed to D arrives at PE2 from CE2, the packet is identified by, for instance, its incoming interface as coming from VPN A. PE2 looks in its per-VPN FIB for VPN A and sees that the next hop is PE1. This is the intra-VPN case.

When a packet addressed to D arrives at PE3 from CE3, the packet is identified, again possibly by virtue of its incoming interface, as coming from VPN B. PE3 looks in

its per-VPN FIB for VPN B and sees that the next hop is PE2. The packet then gets sent to PE2, which sends it on to CE2. CE2 runs the packet through the firewall, and CE2 attempts to forward the packet if the firewall does not reject it. Since the destination is not on-site, the packet gets sent to PE2. This time PE2 identifies the packet as coming from

5   VPN A. PE2 looks up D in its per-VPN FIB for VPN A, and sees that PE1 is the next hop. The packet is then sent to PE1.

In short, when PE router receives a packet from a CE router, it can always identify the CE router from which the packet was just transmitted, so it can identify the VPN from which it just came. This enables the PE router to select the proper per-VPN FIB.

10   Although CE2 ran the packet that it received in the above scenario through the firewall, it would ordinarily be preferred that only packets from outside VPN A receive this treatment, in which case CE2 will need to know whether a packet that it receives is from a different VPN. The way in which this is accomplished is in general a local-configuration matter, but the most-common approach would likely be for CE2 to have

15   two channels to PE2. Suppose, for instance, that CE2 has two different CE2 interfaces for such communication. It would run BGP on both interfaces. On one of the interfaces, it would advertise reachability to some set of addresses in VPN A (including D) and possibly specify appropriate community attributes to ensure that this information is exported to VPN B. PE2 would use VPN A's external VPN ID for information received over this

20   BGP connection. On the other interface, it would advertise reachability to its on-site addresses, and PE2 would use VPN A's internal VPN ID for information received over this BGP connection.

Although the use of different interfaces would be the most-typical way to provide the different channels by which the achieve the internal- and external-route information

25   and traffic are distinguished, internal routes and external routes could be mapped to the same interface, too, with the demultiplexing provided by, say, the presence or absence of cryptographic information in the packet header.

Alternatives

The foregoing discussion describes an advantageous approach to implementing the present invention's teachings, one whose advantages extend not only to situations in which the customer VPNs' address spaces overlap. But the particular approach there de-

5   scribed is far from the only one that can implement the present invention's teachings. For example, some of the routing could be set statically rather than in response to routing protocols such as BGP. Also, although we have described VPN-specific information as being stored in separate tables because that approach seems most convenient, there is no reason in principle why a common table containing VPN-identifying entries could not be

10   used instead.

And our focus on tag switching should not be interpreted to mean that the present invention's teachings are so limited. For instance, although we use tags to contain both the egress-router routing information and the egress-channel routing information, one could instead use, say, encapsulated IP to hide the inner, egress-channel (and thereby

15   VPN-distinguishing) routing information from the transit routers, this is by no means a requirement. We prefer tag switching because it tends to be more efficient, to use less overhead, and to lend itself to uses where the network administration controls the routes to a greater degree than dynamic IP routing ordinarily allows. Also, unlike encapsulated IP, tag switching supports arrangements in which different VPN sites are attached to the

20   networks of different autononmous service-providers that use BGP to exchange routing information and together form the back-bone-providing service-provider network. And tag switching lends itself to applications in which part of the backbone is an ATM link: tags can be put in the ATM header's VCI field.

But even when tags are used, they can represent the exterior-routing information

25   in a way different from the one that the illustrated embodiment employs. For example, although the illustrated embodiment interprets the exterior-routing tag exemplified by T3 to specify a next hop, it could instead simply contain, say, a VPN identifier that the egress router uses to disambiguate the regular IP address.

Although we prefer to use tags for both the egress-router and egress-channel

30   fields, moreover, the applicability of the present invention's teachings is not so limited.

In an architecture in which every PE router always uses different interfaces for links to different VPNs' nodes, for example, the internal-routing field could be provided simply as a tag associated with such an interface. That is, there would be no separate tag for the egress router's interface with the previous P router. In such an arrangement, edge routers

5 could use IGP to install host routes to all of their interfaces with client edge routers. To advertise external reachability, PE2, for instance, would use BGP to specify the IP address of the interface between PE2 and CE2 as the next-hop address for VPN-IPv4 addresses reachable through CE2. And PE2, P2, and P1 would all use TDP to bind tags to the host route to that interface; PE2 would not use the distinguished tag value meaning

10 "pop the tag stack."

In short, the present invention's advantages can be obtained from a wide variety of embodiments. It therefore constitutes a significant advance in the art.

The following four sections, referred to above, set forth the contents of respective Internet drafts in which techniques that embodiments of the present invention may em-

15 ploy are described.

INTERNET DRAFT: LABEL STACK ENCODINGS

Network Working Group	Eric C. Rosen
Internet Draft	Yakov Rekhter
Expiration Date: January 1998	Daniel Tappan
	Dino Farinacci
	Guy Fedorkow
	Cisco Systems, Inc.
	Tony Li
	Juniper Networks, Inc.
	Alex Conta
	Lucent Technologies
	July 1997

Label Switching: Label Stack Encodings

draft-rosen-tag-stack-03.txt

5   Status of this Memo

This document is an Internet-Draft. Internet-Drafts
are working documents of the Internet Engineering Task
Force (IETF), its areas, and its working groups. Note
that other groups may also distribute working docu-
10   ments as Internet-Drafts.

Internet-Drafts are draft documents valid for a maxi-
mum of six months and may be updated, replaced, or ob-
soleted by other documents at any time. It is inap-
propriate to use Internet-Drafts as reference material
15   or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft,
please check the "1id-abstracts.txt" listing contained
in the Internet-Drafts Shadow Directories on
ftp.is.co.za (Africa), nic.nordu.net (Europe), mun-
20   nari.oz.au (Pacific Rim), ds.internic.net (US East
Coast), or ftp.isi.edu (US West Coast).

Abstract

"Multi-Protocol Label Switching (MPLS)" [1,2,9] re-
25   quires a set of procedures for augmenting network
layer packets with "label stacks" (sometimes called
"tag stacks"), thereby turning them into "labeled
packets". Routers which support MPLS are known as
"Label Switching Routers", or "LSRs". In order to
30   transmit a labeled packet on a particular data link,
an LSR must support an encoding technique which, given

a label stack and a network layer packet, produces a labeled packet. This document specifies the encoding to be used by an LSR in order to transmit labeled packets on PPP data links and on LAN data links. This document also specifies rules and procedures for processing the various fields of the label stack encoding.

1. Introduction

"Multi-Protocol Label Switching (MPLS)" [1,2,9] requires a set of procedures for augmenting network layer packets with "label stacks" (sometimes called "tag stacks"), thereby turning them into "labeled packets". Routers which support MPLS are known as "Label Switching Routers", or "LSRs". In order to transmit a labeled packet on a particular data link, an LSR must support an encoding technique which, given a label stack and a network layer packet, produces a labeled packet.

This document specifies the encoding to be used by an LSR in order to transmit labeled packets on PPP data links and on LAN data links.

This document also specifies rules and procedures for processing the various fields of the label stack encoding. Since MPLS is independent of any particular network layer protocol, the majority of such procedures are also protocol-independent. A few, however, do differ for different protocols. In this document, we specify the protocol-independent procedures, and we specify the protocol-dependent procedures for IPv4.

LSRs that are implemented on certain switching de-
vices (such as ATM switches) may use different encod-
ing techniques for encoding the top one or two en-
tries of the label stack. When the label stack has

5 additional entries, however, the encoding technique
described in this document may be used for the addi-
tional label stack entries.

1.1. Specification of Requirements

10 In this document, several words are used to sig-
nify the requirements of the specification. These
words are often capitalized.

MUST

This word, or the adjective "required", means that

15 the definition is an absolute requirement of the
specification.

MUST NOT

This phrase means that the definition is an abso-
lute prohibition of the specification.

20 SHOULD

This word, or the adjective "recommended", means
that there may exist valid reasons in particular cir-
cumstances to ignore this item, but the full implica-
tions must be understood and carefully weighed before

25 choosing a different course.

MAY

This word, or the adjective "optional", means that
this item is one of an allowed set of alternatives.
An implementation which does not include this option

30 MUST be prepared to interoperate with another imple-
mentation which does include the option.

2. The Label Stack

2.1. Encoding the Label Stack

On both PPP and LAN data links, the label stack is represented as a sequence of "label stack entries". Each label stack entry is represented by 4 octets. This is shown in Figure 10.

The label stack entries appear AFTER the data link layer headers, but BEFORE any network layer headers. The top of the label stack appears earliest in the packet, and the bottom appears latest. The network layer packet immediately follows the label stack entry which has the S bit set.

Each label stack entry is broken down into the following fields:

1. Bottom of Stack (S)

    This bit is set to one for the last entry in the label stack (i.e., for the bottom of the stack), and zero for all other label stack entries.

2. Time to Live (TTL)

    This eight-bit field is used to encode a time-to-live value. The processing of this field is described in section 2.3.

3. Class of Service (CoS)

    This three-bit field is used to identify a "Class of Service". The setting of this field is intended to affect the scheduling and/or discard algorithms which are applied to the packet as it is transmitted through the network.

When an unlabeled packet is initially labeled, the value assigned to the CoS field in the label stack entry is determined by policy. Some possible policies are:

- the CoS value is a function of the IP ToS value

- the CoS value is a function of the packet's input interface

- the CoS value is a function of the "flow type"

Of course, many other policies are also possible.

When an additional label is pushed onto the stack of a packet that is already labeled:

- in general, the value of the CoS field in the new top stack entry should be equal to the value of the CoS field of the old top stack entry;

- however, in some cases, most likely at boundaries between network service providers, the value of the CoS field in the new top stack entry may be determined by policy.

4. Label Value

This 20-bit field carries the actual value of the Label.

When a labeled packet is received, the label
value at the top of the stack is looked up. As
a result of a successful lookup one learns:

    (a) information needed to forward the
        packet, such as the next hop and the out-
        going data link encapsulation; however,
        the precise queue to put the packet on,
        or information as to how to schedule the
        packet, may be a function of both the la-
        bel value AND the CoS field value;

    (b) the operation to be performed on the
        label stack before forwarding; this op-
        eration may be to replace the top label
        stack entry with another, or to pop an
        entry off the label stack, or to replace
        the top label stack entry and then to
        push one or more additional entries on
        the label stack.

There are several reserved label values:

    i. A value of 0 represents the "IPv4 Ex-
       plicit NULL Label". This label value is
       only legal when it is the sole label
       stack entry. It indicates that the la-
       bel stack must be popped, and the for-
       warding of the packet must then be
       based on the IPv4 header.

    ii. A value of 1 represents the "Router
       Alert Label". This label value is le-
       gal anywhere in the label stack except

at the bottom. When a received packet
contains this label value at the top of
the label stack, it is delivered to a
local software module for processing.
The actual forwarding of the packet is
determined by the label beneath it in
the stack. However, if the packet is
forwarded further, the Router Alert La-
bel should be pushed back onto the la-
bel stack before forwarding. The use
of this label is analogous to the use
of the "Router Alert Option" in IP
packets [6]. Since this label cannot
occur at the bottom of the stack, it is
not associated with a particular net-
work layer protocol.

iii. A value of 2 represents the "IPv6 Ex-
plicit NULL Label". This label value
is only legal when it is the sole label
stack entry. It indicates that the la-
bel stack must be popped, and the for-
warding of the packet must then be
based on the IPv6 header.

iv. A value of 3 represents the "Implicit
NULL Label". This is a label that an
LSR may assign and distribute, but
which never actually appears in the en-
capsulation. When an LSR would other-
wise replace the label at the top of

the stack with a new label, but the new
label is "Implicit NULL", the LSR will
pop the stack instead of doing the re-
placement. Although this value may
never appear in the encapsulation, it
needs to be specified in the Label Dis-
tribution Protocol, so a value is re-
served.

v. Values 4-16 are reserved.

2.2. Determining the Network Layer Protocol

When the last label is popped from the label stack, it is necessary to de-
termine the particular network layer protocol which is
being carried. Note that the label stack entries
carry no explicit field to identify the network layer
header. Rather, this must be inferable from the value
of the label which is popped from the bottom of the
stack. This means that when the first label is pushed
onto a network layer packet, the label must be one
which is used ONLY for packets of a particular network
layer. Furthermore, whenever that label is replaced
by another label value during a packet's transit, the
new value must also be one which is used only for
packets of that network layer.

2.3. Processing the Time to Live Field

2.3.1. Definitions

The "incoming TTL" of a labeled packet is defined to be the value of the TTL field of the top label stack entry when the packet is received.

The "outgoing TTL" of a labeled packet is defined to be the larger of:

(a) one less than the incoming TTL,

(b) zero.

2.3.2. Protocol-independent rules

If the outgoing TTL of a labeled packet is 0, then the labeled packet MUST NOT be further forwarded; the packet's lifetime in the network is considered to have expired.

Depending on the label value in the label stack entry, the packet MAY be silently discarded, or the packet MAY have its label stack stripped off, and passed as an unlabeled packet to the ordinary processing for network layer packets which have exceeded their maximum lifetime in the network. However, even if the label stack is stripped, the packet MUST NOT be further forwarded.

When a labeled packet is forwarded, the TTL field of the label stack entry at the top of the label stack must be set to the outgoing TTL value.

Note that the outgoing TTL value is a function solely of the incoming TTL value, and is independent of whether any labels are pushed or popped before forwarding. There is no significance to the value of the TTL field in any label stack entry which is not at the top of the stack.

2.3.3. IP-dependent rules

We define the "IP TTL" field to be the value of the IPv4 TTL field, or the value of the IPv6 Hop Limit field, whichever is applicable.

When an IP packet is first labeled, the TTL field of the label stack entry MUST BE set to the value of the IP TTL field. (If the IP TTL field needs to be decremented, as part of the IP processing, it is assumed that this has already been done.)

When a label is popped, and the resulting label stack is empty, then the value of the IP TTL field MUST BE replaced with the outgoing TTL value, as defined above. In IPv4 this also requires modification of the IP header checksum.

3. Fragmentation and Path MTU Discovery

Just as it is possible to receive an unlabeled IP datagram which is too large to be transmitted on its output link, it is possible to receive a labeled packet which is too large to be transmitted on its output link.

It is also possible that a received packet (labeled or unlabeled) which was originally small enough to be transmitted on that link becomes too large by virtue of having one or more additional labels pushed onto its label stack. In label switching, a packet may grow in size if additional labels get pushed on. Thus if one receives a labeled packet with a 1500-byte frame payload, and pushes on an additional label, one needs to forward it as frame with a 1504-byte payload.

This section specifies the rules for processing labeled packets which are "too large". In particular, it provides rules which ensure that hosts implementing RFC 1191 Path MTU Discovery, and hosts using IPv6, will be able to generate IP datagrams that do not need fragmentation, even if they get labeled as the traverse the network.

In general, hosts which do not implement RFC 1191 Path MTU Discovery send IP datagrams which contain no more than 576 bytes. Since the MTUs in use on most data links today are 1500 bytes or more, the probability that such datagrams will need to get fragmented, even if they get labeled, is very small.

Some hosts that do not implement RFC 1191 Path MTU Discovery will generate IP datagrams containing 1500 bytes, as long as the IP Source and Destination addresses are on the same subnet. These datagrams will not pass through routers, and hence will not get fragmented.

Unfortunately, some hosts will generate IP datagrams containing 1500 bytes, as long the IP Source and Destination addresses do not have the same classful network number. This is the one case in which there is any risk of fragmentation when such datagrams get labeled. (Even so, fragmentation is not likely unless the packet must traverse an ethernet of some sort between the time it first gets labeled and the time it gets unlabeled.)

This document specifies procedures which allow one to configure the network so that large datagrams from hosts which do not implement Path MTU Discovery get

fragmented just once, when they are first labeled. These procedures make it possible (assuming suitable configuration) to avoid any need to fragment packets which have already been labeled.

3.1. Terminology

With respect to a particular data link, we can use the following terms:

- Frame Payload:

    The contents of a data link frame, excluding any data link layer headers or trailers (e.g., MAC headers, LLC headers, 802.1Q or 802.1p headers, PPP header, frame check sequences, etc.).

    When a frame is carrying an an unlabeled IP datagram, the Frame Payload is just the IP datagram itself. When a frame is carrying a labeled IP datagram, the Frame Payload consists of the label stack entries and the IP datagram.

- Conventional Maximum Frame Payload Size:

    The maximum Frame Payload size allowed by data link standards. For example, the Conventional Maximum Frame Payload Size for ethernet is 1500 bytes.

- True Maximum Frame Payload Size:

    The maximum size frame payload which can be sent and received properly by the interface hardware attached to the data link.

    On ethernet and 802.3 networks, it is believed that the True Maximum Frame Payload Size is 4-8 bytes larger than the Conventional Maximum Frame Payload Size (as long neither an 802.1Q header

nor an 802.1p header is present, and as long as
neither can be added by a switch or bridge while
a packet is in transit to its next hop). For ex-
ample, it is believed that most ethernet equip-
ment could correctly send and receive packets
carrying a payload of 1504 or perhaps even 1508
bytes, at least, as long as the ethernet header
does not have an 802.1Q or 802.1p field.

On PPP links, the True Maximum Frame Payload
Size may be virtually unbounded.

- Effective Maximum Frame Payload Size for Labeled
  Packets:
  This is either be the Conventional Maximum Frame
  Payload Size or the True Maximum Frame Payload
  Size, depending on the capabilities of the equip-
  ment on the data link and the size of the
  ethernet header being used.

- Initially Labeled IP Datagram
  Suppose that an unlabeled IP datagram is received
  at a particular LSR, and that the the LSR pushes
  on a label before forwarding the datagram. Such
  a datagram will be called an Initially Labeled IP
  Datagram at that LSR.

- Previously Labeled IP Datagram
  An IP datagram which had already been labeled be-
  fore it was received by a particular LSR.

3.2. Maximum Initially Labeled IP Datagram Size
  Every LSR which is capable of
  (a) receiving an unlabeled IP datagram,
  (b) adding a label stack to the datagram, and

(c) forwarding the resulting labeled packet,

MUST support a configuration parameter known as the "Maximum IP Datagram Size for Labeling", which can be set to a non-negative value.

If this configuration parameter is set to zero, it has no effect.

If it is set to a positive value, it is used in the following way. If:

(a) an unlabeled IP datagram is received, and

(b) that datagram does not have the DF bit set in its IP header, and

(c) that datagram needs to be labeled before being forwarded, and

(d) the size of the datagram (before labeling) exceeds the value of the parameter, then

(a) the datagram must be broken into fragments, each of whose size is no greater than the value of the parameter, and

(b) each fragment must be labeled and then forwarded.

If this configuration parameter is set to a value of 1488, for example, then any unlabeled IP datagram containing more than 1488 bytes will be fragmented before being labeled. Each fragment will be capable of being carried on a 1500-byte data link, without further fragmentation, even if as many as three labels are pushed onto its label stack.

In other words, setting this parameter to a non-zero value allows one to eliminate all fragmentation of Previously Labeled IP Datagrams, but it may cause some

unnecessary fragmentation of Initially Labeled IP Da-
tagrams.

Note that the parameter has no effect on IP Da-
tagrams that have the DF bit set, which means that it
has no effect on Path MTU Discovery.

3.3. When are Labeled IP Datagrams Too Big?

A labeled IP datagram whose size exceeds the Conven-
tional Maximum Frame Payload Size of the data link
over which it is to be forwarded MAY be considered to
be "too big".

A labeled IP datagram whose size exceeds the True
Maximum Frame Payload Size of the data link over which
it is to be forwarded MUST be considered to be "too
big".

A labeled IP datagram which is not "too big" MUST be
transmitted without fragmentation.

3.4. Processing Labeled IPv4 Datagrams which are Too Big

If a labeled IPv4 datagram is "too big", and the DF
bit is not set in its IP header, then the LSR MAY dis-
card the datagram.

Note that discarding such datagrams is a sensible
procedure only if the "Maximum Initially Labeled IP
Datagram Size" is set to a non-zero value in every LSR
in the network which is capable of adding a label
stack to an unlabeled IP datagram.

If the LSR chooses not to discard a labeled IPv4 da-
tagram which is too big, or if the DF bit is set in
that datagram, then it MUST execute the following al-
gorithm:

1. Strip off the label stack entries to obtain the
   IP datagram.
2. Let N be the number of bytes in the label stack
   (i.e, 4 times the number of label stack en-
   tries).
3. If the IP datagram does NOT have the "Don't
   Fragment" bit set in its IP header:
   a. convert it into fragments, each of which
      MUST be at least N bytes less than the Ef-
      fective Maximum Frame Payload Size.
   b. Prepend each fragment with the same label
      header that would have been on the origi-
      nal datagram had fragmentation not been
      necessary.
   c. Forward the fragments

4. If the IP datagram has the "Don't Fragment" bit
   set in its IP header:
   a. the datagram MUST NOT be forwarded
   b. Create an ICMP Destination Unreachable
      Message:
      i. set its Code field (RFC 792) to
         "Fragmentation Required and DF
         Set",

      ii. set its Next-Hop MTU field (RFC
          1191) to the difference between the Effective
          Maximum Frame Payload Size and the
          value of N

    c. If possible, transmit the ICMP Destina-
tion Unreachable Message to the source of
the of the discarded datagram.

3.5. Processing Labeled IPv6 Datagrams which are Too Big

    To process a labeled IPv6 datagram which is too big,
an LSR MUST execute the following algorithm:

1. Strip off the label stack entries to obtain the
   IP datagram.
2. Let N be the number of bytes in the label stack
   (i.e, 4 times the number of label stack en-
   tries).
3. If the IP datagram contains more than 576 bytes
   (not counting the label stack entries), then:
       a. Create an ICMP Packet Too Big Message,
   and set its Next-Hop MTU field to the
   difference between the Effective Maximum
   Frame Payload Size and the value of N
       b. If possible, transmit the ICMP Packet Too
   Big Message to the source of the da-
   tagram.
       c. discard the labeled IPv6 datagram.
4. If the IP datagram is not larger than 576 oc-
   tets, then
       a. Convert it into fragments, each of which
   MUST be at least N bytes less than the Effective
   Maximum Frame Payload Size.
       b. Prepend each fragment with the same label
   header that would have been on the origi-

nal datagram had fragmentation not been necessary.

    c. Forward the fragments.

Reassembly of the fragments will be done at the destination host.

3.6. Implications with respect to Path MTU Discovery

The procedures described above for handling datagrams which have the DF bit set, but which are "too large", have an impact on the Path MTU Discovery procedures of RFC 1191. Hosts which implement these procedures will discover an MTU which is small enough to allow n labels to be pushed on the datagrams, without need for fragmentation, where n is the number of labels that actually get pushed on along the path currently in use.

In other words, datagrams from hosts that use Path MTU Discovery will never need to be fragmented due to the need to put on a label header, or to add new labels to an existing label header. (Also, datagrams from hosts that use Path MTU Discovery generally have the DF bit set, and so will never get fragmented anyway.)

However, note that Path MTU Discovery will only work properly if, at the point where a labeled IP Datagram's fragmentation needs to occur, it is possible to route to the packet's source address. If this is not possible, then the ICMP Destination Unreachable message cannot be sent to the source.

3.6.1. Tunneling through a Transit Routing Domain

Suppose one is using MPLS to "tunnel" through a transit routing domain, where the external routes are not leaked into the domain's interior routers. If a packet needs fragmentation at some router within the domain, and the packet's DF bit is set, it is necessary to be able to originate an ICMP message at that router and have it routed correctly to the source of the fragmented packet. If the packet's source address is an external address, this poses a problem.

Therefore, in order for Path MTU Discovery to work, any routing domain in which external routes are not leaked into the interior routers MUST have a default route which causes all packets carrying external destination addresses to be sent to a border router. For example, one of the border routers may inject "default" into the IGP.

3.6.2. Tunneling Private Addresses through a Public Backbone

In other cases where MPLS is used to tunnel through a routing domain, it may not be possible to route to the source address of a fragmented packet at all. This would be the case, for example, if the IP addresses carried in the packet were private addresses, and MPLS were being used to tunnel those packets through a public backbone.

In such cases, the LSR at the transmitting end of the tunnel MUST be able to determine the MTU of the tunnel as a whole. It SHOULD do this by sending packets through the tunnel to the tunnel's receiving endpoint, and performing Path MTU Discovery with those

packets. Then any time the transmitting endpoint of the tunnel needs to send a packet into the tunnel, and that packet has the DF bit set, and it exceeds the tunnel MTU, the transmitting endpoint of the tunnel MUST send the ICMP Destination Unreachable message to the source, with code "Fragmentation Required and DF Set", and the Next-Hop MTU Field set as described above.

4. Transporting Labeled Packets over PPP

The Point-to-Point Protocol (PPP) [7] provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols for establishing and configuring different network-layer protocols.

This section defines the Network Control Protocol for establishing and configuring label Switching over PPP.

4.1. Introduction

PPP has three main components:

1. A method for encapsulating multi-protocol datagrams.

2. A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.

3. A family of Network Control Protocols for establishing and configuring different network-layer protocols.

In order to establish communications over a point-to-point link, each end of the PPP link must first send LCP packets to configure and test the data link. After the link has been established and optional facilities have been negotiated as needed by the LCP, PPP must send "MPLS Control Protocol" packets to enable the transmission of labeled packets. Once the "MPLS Control Protocol" has reached the Opened state, labeled packets can be sent over the link.

The link will remain configured for communications until explicit LCP or MPLS Control Protocol packets close the link down, or until some external event occurs (an inactivity timer expires or network administrator intervention).

4.2. A PPP Network Control Protocol for MPLS

The MPLS Control Protocol (MPLSCP) is responsible for enabling and disabling the use of label switching on a PPP link. It uses the same packet exchange mechanism as the Link Control Protocol (LCP). MPLSCP packets may not be exchanged until PPP has reached the Network-Layer Protocol phase. MPLSCP packets received before this phase is reached should be silently discarded.

The MPLS Control Protocol is exactly the same as the Link Control Protocol [7] with the following exceptions:

1. Frame Modifications

   The packet may utilize any modifications to the basic frame format which have been negotiated during the Link Establishment phase.

2. Data Link Layer Protocol Field

Exactly one MPLSCP packet is encapsulated in the
PPP Information field, where the PPP Protocol
field indicates type hex 8081 (MPLS).

3. Code field

Only Codes 1 through 7 (Configure-Request, Con-
figure-Ack, Configure-Nak, Configure-Reject,
Terminate-Request, Terminate-Ack and Code-
Reject) are used. Other Codes should be treated
as unrecognized and should result in Code-
Rejects.

4. Timeouts

MPLSCP packets may not be exchanged until PPP
has reached the Network-Layer Protocol phase.
An implementation should be prepared to wait for
Authentication and Link Quality Determination to
finish before timing out waiting for a Config-
ure-Ack or other response. It is suggested that
an implementation give up only after user inter-
vention or a configurable amount of time.

5. Configuration Option Types

None.

4.3. Sending Labeled Packets

Before any labeled packets may be communicated, PPP
must reach the Network-Layer Protocol phase, and the
MPLS Control Protocol must reach the Opened state.

Exactly one labeled packet is encapsulated in the
PPP Information field, where the PPP Protocol field
indicates either type hex 0081 (MPLS Unicast) or type
hex 0083 (MPLS Multicast). The maximum length of a

labeled packet transmitted over a PPP link is the same
as the maximum length of the Information field of a
PPP encapsulated packet.

The format of the Information field itself is as de-
fined in section 2.

Note that two codepoints are defined for labeled
packets; one for multicast and one for unicast. Once
the MPLSCP has reached the Opened state, both label
Switched multicasts and label Switched unicasts can be
sent over the PPP link.

4.4. Label Switching Control Protocol Configuration Op-
tions

There are no configuration options.

5. Transporting Labeled Packets over LAN Media

Exactly one labeled packet is carried in each frame.

The label stack entries immediately precede the net-
work layer header, and follow any data link layer
headers, including any VLAN headers, 802.1p headers,
and/or 802.1Q headers that may exist.

The ethertype value 8847 hex is used to indicate
that a frame is carrying an MPLS unicast packet.

The ethertype value 8848 hex is used to indicate
that a frame is carrying an MPLS multicast packet.

These ethertype values can be used with either the
ethernet encapsulation or the 802.3 SNAP/SAP encapsu-
lation to carry labeled packets.

6. Security Considerations

Security considerations are not discussed in this document.

7. Authors' Addresses

Eric C. Rosen Cisco Systems, Inc. 250 Apollo Drive Chelmsford, MA, 01824 E-mail: erosen@cisco.com

Dan Tappan Cisco Systems, Inc. 250 Apollo Drive Chelmsford, MA, 01824 E-mail: tappan@cisco.com

Dino Farinacci Cisco Systems, Inc. 170 Tasman Drive San Jose, CA, 95134 E-mail: dino@cisco.com

Yakov Rekhter Cisco Systems, Inc. 170 Tasman Drive San Jose, CA, 95134 E-mail: yakov@cisco.com

Guy Fedorkow Cisco Systems, Inc. 250 Apollo Drive Chelmsford, MA, 01824 E-mail: fedorkow@cisco.com

Tony Li Juniper Networks 3260 Jay Street Santa Clara, CA 95051 E-mail: tli@jnx.com

Alex Conta Lucent Technologies 300 Baker Avenue Concord, MA, 01742 E-mail: aconta@lucent.com

8. References

[1] "Tag Switching Architecture - Overview", 1/9/97, draft-rekhter- tagswitch-arch-00.txt, Rekhter, Davie, Katz, Rosen, Swallow

[2] "A Framework for Multiprotocol Label Switching", 5/12/97, draft- ietf-mpls-framework-00.txt, Callon, Doolan, Feldman, Fredette, Swallow, Visanathawan

[3] "Internet Protocol", RFC 791, 9/81, Postel

[4] "Internet Control Message Protocol", RFC 792, 9/81, Postel

[5] "Path MTU Discovery", RFC 1191, 11/90, Mogul & Deering

[6] "IP Router Alert Option", RFC 2113, 2/97, Katz

[7] "The Point-to-Point Protocol (PPP)", RFC 1661, 7/94, Simpson

[8] "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 1885, 12/95, Conta, Deering

[9], "A Proposed Architecture for MPLS", 7/97, draft-rosen-mpls- arch-00.txt, Rosen, Viswanathan, Callon

INTERNET DRAFT: TAG DISTRIBUTION PROTOCOL

Network Working Group   P Doolan

Internet Draft   cisco Systems

Expiration Date: November 1997

                                                 B Davie

                                               cisco Systems

                                                  D Katz

                                              Juniper Networks

                                                 Y Rekhter

                                               cisco Systems

                                                 E Rosen

                                               cisco Systems

                                                 May 1997

Tag Distribution Protocol

draft-doolan-tdp-spec-01.txt

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note

81

that other groups may also distribute working docu-
ments as Internet-Drafts.

Internet-Drafts are draft documents valid for a
maximum of six months and may be updated, replaced, or
obsoleted by other documents at any time. It is inap-
propriate to use Internet-Drafts as reference material
or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft,
please check the "1id-abstracts.txt" listing contained
in the Internet-Drafts Shadow Directories on
ftp.is.co.za (Africa), nic.nordu.net (Europe), mun-
nari.oz.au (Pacific Rim), ds.internic.net (US East
Coast), or ftp.isi.edu (US West Coast).

1. Abstract

An overview of a tag switching architecture is pro-
vided in [Rekhter]. This document defines the Tag
Distribution Protocol (TDP) referred to in [Rekhter].

TDP is a two party protocol that runs over a connec-
tion oriented transport layer with guaranteed sequen-
tial delivery. Tag Switching Routers use TDP to com-
municate tag binding information to their peers. TDP
supports multiple network layer protocols including
but not limited to IPv4, IPv6, IPX and AppleTalk.

We define here the PDUs and operational procedures
for this TDP and specify its transport requirements.
We also define aspects of the protocol that are spe-
cific to the case where it is run over an ATM data-
link.

2. Protocol Overview

A tag switching architecture is described in [Rek-hter]. As explained in that document Tag Switching Routers (TSRs) create tag bindings, and then distribute the tag binding information among other TSRs.

TDP provides the means for TSRs to distribute, request, and release tag binding information for multiple network layer protocols. TDP also provides means to open, monitor and close TDP sessions and to indicate errors that occur during those sessions.

TDP is a two party protocol that requires a connection oriented transport layer with guaranteed sequential delivery. We use TCP as the transport for TDP.

A TSR that wishes to exchange tag bindings with another opens a TCP connection to the TDP port (TBD) on that other TSR. Once the TCP connection has been established then the TSRs exchange TDP PDUs that encode tag binding information. TDP is symmetrical in that once the TCP connection has been opened the peer TSRs may each send and receive TDP PDUs at will.

A single TSR may have TDP sessions with multiple other TSRs. Each of these sessions is completely independent of the others. Multiple TDP sessions may exist between any given pair of TSRs. Each of these sessions is completely independent of the others. TDP sessions are identified by the 'TDP Identifier' field in the TDP header (see below).

TDP does not require any keepalive notification from the transport, but implements its own keepalive timer. The usage is straightforward: peers must communicate within the period specified by the timer. Each time a TDP peer receives a TDP PDU it resets the

timer. If the timer expires some number of times without reception of a TDP PDU from the remote system the TDP closes the session with its peer.

When a TSR determines that it lost a TDP session with another TSR, if the TSR has any tag bindings that were created as a result of receiving tag binding requests from the peer, the TSR may destroy these bindings (and deallocate tags associated with these binding).

When a TSR determines that it lost a TDP session with another TSR, the TSR shall no longer use the binding information it received from the other TSR.

The procedures that govern when other components in a TSR invoke services from TDP and how a TSR maintains its TIBs are beyond the scope of this document.

The use of TDP does not preclude the use of other mechanisms to distribute tag binding information.

2.1. TDP and Tagswitching over ATM

The tagswitching architecture [Rekhter] describes application of tag switching to ATM, [Davie] provides more details and describes a number of features of TDP required specifically to support this ATM case. We describe control circuit useage and encapsulation here. The sections on TDP_PIE_BIND and TDP_PIE_REQUEST_BIND describe how 'Hop Count' referred to in [Davie] is carried.

2.1.1. Default VPI/VCI

By default the TDP connection between two ATM-TSRs uses VPI/VCI 0/32. The default TDP connection uses the

LLC/SNAP encapsulation defined in RFC1483 [Heinanen]. This TDP VC may be used to exchange other LLC/SNAP encapsulated traffic. In particular the TDP VC might be used to carry Network Layer routing information.

5 There are circumstances (see ATM_TAG_RANGE) when this VC is also used to carry data traffic.

TDP provides means to advertise the range of, and negotiate the encapsulation used on, the data VCs. See the section on TDP_PIE_OPEN for further details.

10 Cooperating TSRs may agree to use VPI/VCI other than 0/32 as the TDP VC, how they do this (management) is outside the scope of this document.

3. State machines

15 We describe the TDP's behavior in terms of a state machine. We define the TDP state machine to have four possible states and present the behavior as a state transition table and diagram.

3.1. TDP state transition table

20

STATE	EVENT	NEW STATE
	Initialization	INITIALIZED
INITIALIZED	Sent TDP_PIE_OPEN	OPENSENT
	Received TDP_PIE_OPEN	OPENREC
OPENREC	Received TDP_PIE_KEEP_ALIVE	OPERATIONAL
	Received Any other TDP PDU	INITIALIZED
OPENSENT	Received TDP_PIE_OPEN & Transmit TDP_PIE_KEEP_ALIVE	OPENREC

```
        Received Any other TDP PDU  INITIALIZED
        Sent TDP_PIE_NOTIFICATION    INITIALIZED


OPERATIONAL Rx/Tx TDP_PIE_NOTIFICATION
                with CLOSING parameter INITIALIZED
        Other TDP PDUs               OPERATIONAL
        Timeout                      INITIALIZED
```

3.2. TDP state transition diagram

See Figure 11.

3.3. Transport connections

A TSR that implements TDP opens a TCP connection to a peer TSR. Once open, and regardless of which TSR opened it, the TCP connection is used bidirectionally. That is there is only one TCP 'connection' used for a TDP session between two TSRs. TDP uses TCP port (TBD).

3.4. Timeout

Timeout in the state transition table and diagram indicates that the keep alive timer set to HOLD_TIME has expired. See TDP_PIE_OPEN for a discussion of this mechanism.

4. Protocol Data Units (PDUs)

TDP PDUs are variable length and consist of a fixed header and one or more Protocol Information Elements (PIE) each with a Type Length Value (TLV) structure. Within a single PIE TLVs may be nested to an arbitrary depth.

A single TDP PDU may contain multiple PIEs.   The maximum TDP PDU size is 4096 octets.

4.1. TDP Fixed Header

The fixed header of the TDP PDU is shown in Figure 12.

Version:

> This two octet unsigned integer contains the version number of the protocol. A TDP version number must lie in the range 0x01 <= Version <= 0xFF. This version of the TDP specification specifies protocol Version = 1.

LENGTH:

This two octet integer specifies the length in octets of the data portion of the PDU. LENGTH is set to the length of the PDU in octets minus four.

TDP Identifier:

Six octet unsigned integer containing a unique identifier for the TSR that generated the PDU. The value of this Identifier is determined on startup. The first four octets encode an IP address assigned to the TSR. T he last two octets represent the 'instance' of TDP on the TSR. A TSR with only one active TDP session would supply the value zero in this field.

Res:

This field is reserved. It must be set to zero on transmission and must be ignored on receipt.

4.2. TDP TLVs

The TDP fixed header frames Protocol Information Elements (PIEs) that have a Type Length Value (TLV) structure.

In this protocol TYPE is a 16 bit integer value that encodes how the VALUE field is to be interpreted. Within a single PIE TLVs may be nested to an arbitrary depth. A TDP must silently discard TLVs that it does not recognize.

LENGTH is an unsigned 16 bit integer value that encodes the length of the VALUE field in octets. LENGTH is set to the length of the whole TLV in octets minus four. A LENGTH of zero indicates that there is no value field present.

VALUE is an octet string of length LENGTH octets that encodes information the semantics of which are indicated by the TYPE field.

A single TLV has the following format that Figure 13 shows.

4.3. Example TDP PDU

A complete TDP PDU containing two PIEs having 4 and 5 octets of Value field respectively would have the following structure that Figure 14 depicts:

4.4. PIEs defined in V1 of TDP

The following PIEs are defined for this version of the protocol. They are described in the sections that follow

Type 0x100 TDP_PIE_OPEN

Type 0x200 TDP_PIE_BIND

Type 0x300 TDP_PIE_REQUEST_BIND

Type 0x400 TDP_PIE_WITHDRAW_BIND

Type 0x500 TDP_PIE_KEEP_ALIVE

Type 0x600 TDP_PIE_NOTIFICATION

Type 0x700 TDP_PIE_RELEASE_BIND

Type 0x800 Unassigned

. . . .

Type 0xFF00

Each of these PIEs may have optional TLV encoded parameters.

4.5. TDP_PIE_OPEN

TDP_PIE_OPEN is the first PIE sent by a TSR initiating a TDP session to its peer. It is sent immediately after the TCP connection has been opened. The TSR receiving a TDP_PIE_OPEN responds either with a TDP_PIE_KEEPALIVE or with a TDP_PIE_NOTIFICATION.

4.5.1. Initiating a TDP session

A TSR initiating a TDP session sets the TDP_OPEN_PIE's fields as described below, issues a PDU containing it to the target peer, the TDP state machine transitions to the OPENSENT state.

While in the OPENSENT state a TSR takes the following actions:

If it receives an 'acceptable' TDP_PIE_OPEN then TSR sends a TDP_PIE_KEEPALIVE and the TDP state machine transitions to the OPEN_REC state.

Receipt of any other PDU is an error and results in sending a TDP_PIE_NOTIFICATION indicating a bad open and transition to the INITIALIZED state.

4.5.2. Passive OPEN

A TSR in the INITIALIZED state that receives a TDP_PIE_OPEN behaves as follows:

If it can support the version of the protocol proposed by the TSR that issued the TDP_PIE_OPEN then it sets Version in all its subsequent communication with that TSR to the value proposed in Prop Ver and obeys the rules specified for that version of the protocol.

TSR sends a PDU containing a TDP_PIE_OPEN PIE to the TSR that initiated the TDP session.

TSR sends a PDU containing a TDP_PIE_KEEPALIVE PIE to the TSR that initiated the TDP session.

The TDP state machine transitions to the OPEN_REC state

If the TSR cannot support the version of the protocol proposed in the TDP_PIE_OPEN then it sends a TDP_PIE_NOTIFICATION PDU that informs the TSR which generated the PIE_OPEN of the version(s) it can support. The TDP state machine transitions to the INITIALIZED state. See below under errors for more details.

4.5.3. OPENREC state

When in the OPENREC state a TSR takes the following actions:

If a TDP_PIE_KEEPALIVE is received then it transitions to the OPERATIONAL state.

Receipt of any other PDU causes the generation of a TDP_PIE_NOTIFICATION and transition to the INITIALIZED state.

The TDP_PIE_OPEN has the format that Figure 15 depicts.

TYPE:

Type field as described above. Set to 0x100 for TDP_PIE_OPEN.

LENGTH:

Length in octets of the value field of this PIE. LENGTH is set to the length of the whole PIE in octets minus four.

Prop Ver:

The Version of the TDP that the TSR that generated this PDU proposes be used for this TDP session once it is established. Note that the session is not established until the TSR that issues a TDP_PIE_OPEN receives a TDP_PIE_OPEN in response.

Hold Time:

Two octet unsigned non zero integer that indicates the number of seconds that the peer initiating the connection proposes for the value of the Hold Timer. Upon receipt of a PDU with PIE TDP_PIE_OPEN, a TDP

peer MUST calculate the value of the Hold Timer by
using the smaller of its configured HOLD_TIME and the
HOLD_TIME received in the PDU. The value chosen for
HOLD_TIME indicates the maximum number of seconds
that may elapse between the receipt of successive
PDUs from the TDP peer. The Hold Timer is reset each
time a TDP_PDU arrives. If the timer expires without
the arrival of a TDP_PDU then a TDP_NOTIFICATION with
the optional parameter CLOSING is sent.

Optional Parameters:
This variable length field contains zero or more
optional PIEs supplied in TLV structures.
See Figure 16.

DOWNSTREAM_ON_DEMAND:
A TSR may supply this optional parameter to indi-
cate that it wishes to use downstream tag alloca-
tion on demand. When either of the peers in a TDP
session indicates that it requires downstream allo-
cation on demand then both shall use that mecha-
nism. TSRs operating in downstream on demand pro-
vide bindings only in response to
TDP_PIE_REQUEST_BINDs.

ATM_TAG_RANGE:

An ATM-TSR supplies this parameter to indicate to
its ATM peer the range of VCIs that it can use as
tags (on this VP). An ATM TSR, when satisfying a
TDP_PIE_BIND_REQUEST, may only generate

VCI/prefix bindings, ie bindings of BLIST_TYPE 6, containing VCI values from the range communicated to it using this optional parameter.

If an ATM-TSR is unable to generate a BLIST_TYPE 6 binding within the constraints imposed by ATM_TAG_RANGE it may generate a binding of BLIST_TYPE 2. [In that case the TSR receiving the binding sends data traffic on the default TDP VCI but tagged with the BLIST_TYPE 2 tag]

The value for this optional parameter is a list of entries of the form that Figure 17 depicts

VPI:

32 bit unsigned integer encoding the VPI to the which the following VCI range bounds apply.

VCI Upper range bound:

32 bit unsigned integer encoding the upper bound of a block of VCIs that the ATM_TSR originating the TDP_PIE_OPEN is making available as tags. VCI values between and including Upper and Lower range bound may be used as tags.

VCI Lower range bound:

32 bit unsigned integer encoding the lower bound of a block of VCIs that the ATM_TSR originating the TDP_PIE_OPEN is making available as tags. VCI values between and including Upper and Lower range bound may be used as tags.

The number of entries may be deduced from the value in the Length field. VCI tags may be allocated from the range indicated by the upper/lower values inclusive of those values. There must be at least one entry. There may be more than one.There may be more than one entry with the same VPI value.

ATM_NULL_ENCAPSULATION:

An ATM-TSR supplies this parameter to indicate that it supports the null encapsulation of RFC1483 [Heinanen] for its data VCs. In this case IP packets are carried directly inside AAL5 frames. This option is only used by an ATM-TSR that it is configured to support a single level of tagging. See [Davie] for more details.

An ATM-TSR that cannot support this option will generate the error TDP_WRONG_ENCAPS.

4.5.4. Errors

All Errors generated by the receipt of a TDP_PIE_OPEN are reported by issuing a TDP_PIE_NOTIFICATION. The value field of the PIE contains one or more TLVs describing individual errors with more precision.

```
+-------------------------+-------+--------+----------+
|Error                    | Type  | Length | Value    |
+-------------------------+-------+--------+----------+
|TDP_OPEN_UNSUPPORTED_VER | 0x1F0 | Var    |See below |
+-------------------------+-------+--------+----------+
|TDP_BAD_OPEN             | 0x1F1 |   0    |   0      |
+-------------------------+-------+--------+----------+
|TDP_WRONG_ENCAPS         | 0x1F2 |   0    |   0      |
+-------------------------+-------+--------+----------+
```

4.5.4.1. TDP_OPEN_UNSUPPORTED_VER:

This error is issued to indicate to the TSR that generated the TDP_PIE_OPEN that this TSR does not support the version of TDP proposed in 'Prop Ver' in the PIE_OPEN. TDP_OPEN_UNSUPPORTED_VER reports the version(s) of the protocol that this TSR does support.

A TSR that receives this error may choose to reissue the TDP_PIE_OPEN specifying a version of the protocol that the target systems has indicated it can support. If a TSR is to take this action it should not close (and reopen) the TCP connection before so doing but should leave the connection 'up' during the negotiation process.

A TSR that generates this error should anticipate that the other system may reissue the TDP_PIE_OPEN and should wait at least TRANSPORT_HOLDDOWN seconds (default 30) before it closes the TCP connection. The TRANSPORT_HOLDDOWN timer is started when a TDP_PIE_NOTIFICATION containing TDP_OPEN_UNSUPPORTED_VER is sent and is reset on reception of a TDP_PIE_OPEN. These measure are designed to stop the version negotiation mechanism 'thrashing' the transport setup mechanism.

TYPE:

TDP_OPEN_UNSUPPORTED_VER = 0x1F0

LENGTH:

Length in octets of the value field of this PIE.
LENGTH is set to the length of the whole PIE in oc-
tets minus four.

VALUE:

One or more 2 octet integers that encode the Ver-
sion(s) of the protocol that this TSR supports.

The format of an NOTIFICATION PIE containing
TDP_OPEN_UNSUPPORTED_VER is:

4.5.4.2. TDP_BAD_OPEN

This error is issued to indicate failure during the
open phase.

4.5.4.3. TDP_WRONG_ENCAPS

This error is used to indicate that an ATM-TSR will
not support the null encapsulation proposed in the
TDP_PIE_OPEN (by the inclusion of the option
ATM_NULL_ENCAPSULATION).

4.6. TDP_PIE_BIND

TDP_PIE_BIND is sent from one TSR to another to dis-
tribute tag bindings. Transmission of a TDP_PIE_BIND
may occur as a result of some local decision or it may
be in response to the reception of a TDP_REQUEST_BIND.
This PIE has the following format

TYPE:

Type field as described above. Set to 0x200 for TDP_PIE_BIND.

LENGTH:

Length in octets of the value field of this PIE. LENGTH is set to the length of the whole PIE in octets minus four.

Request ID:

If this TDP_PIE_BIND is generated in response to a TDP_PIE_REQUEST_BIND then TSR places the value of the Request ID from that request PIE in this field. For all other TDP_PIE_BINDS this field must be set to zero.

AFAM:

This 16 bit integer contains a value from ADDRESS FAMILY NUMBERS in Assigned Numbers [Reynolds] that encodes the address family that the network layer address in the tag bindings in the BINDING_LIST is from. This protocol provides support for multiple network address families.

BLIST_TYPE:

This 16 bit integer contains a value from the table below that encodes the format and semantics of the BLIST entries in the BINDING_LIST field.

> BLIST_TYPE   BLIST entry format

```
0        Null list (see TDP_PIE_WITHDRAW_BIND)
1        32 bit Upstream assigned
2        32 bit Downstream assigned
3        32 bit Multicast Upstream assigned (*,G)
4        32 bit Multicast Upstream assigned (S,G)
5        32 bit Upstream assigned VCI tag
6        32 bit Downstream assigned VCI tag
```

The formats are defined below.

BLIST_LENGTH:

Two octet unsigned integer that encodes the length of the BINDING_LIST

BINDING_LIST:

Variable length field consisting of one or more BLIST entries of the type indicated by BLIST_TYPE.

Optional Parameters:

This variable length field contains zero or more optional PIEs supplied in TLV structures.

4.6.1. BLIST_TYPE 0

BLIST_TYPE = 0 indicates that there are no BLIST entries. See TDP_PIE_WITHDRAW_BIND for further details.

4.6.2. BLIST_TYPE 1 and 2

A BLIST_TYPE 1 contains Upstream assigned tags. A TDP must only include tag values in a BLIST_TYPE 1 tag entry that lie between the values, inclusive of those values, that the TSR to whom the TDP_PIE_BIND is being sent indicated it could support during the OPEN phase.

BLIST entries of type 1 and 2 have the format given in Figure 18.

Precedence

8 bit unsigned integer containing the precedence with which traffic bearing this tag will be serviced by the TSR that issued the TDP_PIE_BIND. [Note that the precedence is likely to be restricted to perhaps three bits of the space reserved here.]

Tag:

Tag is a 32 bit unsigned integer encoding the value of the tag.

Pre Len:

This one octet unsigned integer contains the length in bits of the address prefix that follows.

Prefix:

A variable length field containing an address prefix whose length, in bits, was specified in the previous (Pre Len) field. A Prefix is padded with sufficient trailing zero bits to cause the end of the field to fall on an octet boundary.

4.6.3. BLIST_TYPE 3

This binding allows the association of a tag with the (*,G) shared tree. See [Deering] for a discussion of (*,G) shared trees.

The (*,G) binding has the format that Figure 19 illustrates.

Precedence

8 bit unsigned integer containing the precedence
with which traffic bearing this tag will be serviced
by the TSR that issued the TDP_PIE_BIND. [Note that
the precedence is likely to be restricted to perhaps
three bits of the space reserved here.]

Tag:

Tag is a 32 bit unsigned integer encoding the value
of the tag.

Multicast Group Address G:

Multicast Group Address. The length of this address
is network layer specific and can be deduced from
the value of AFAM. The diagram above illustrates a
four octet IPv4 address format.

4.6.4. BLIST_TYPE 4

This binding type allows association of a tag with
a (S,G) source rooted tree. See [Deering] for a dis-
cussion of (S,G) trees.

The (S,G) binding has the format that Figure 20 de-
picts.

Precedence

8 bit unsigned integer containing the precedence
with which traffic bearing this tag will be serviced
by the TSR that issued the TDP_PIE_BIND. [Note that
the precedence is likely to be restricted to perhaps
three bits of the space reserved here.]

Tag:

Tag is a 32 bit unsigned integer encoding the value
of the tag.

Source Address S:

Network Layer address of the source sending to the
G tree. The length of this address is network layer
specific and can be deduced from the value of AFAM.
The diagram above illustrates a four octet IPv4 ad-
dress format.

Multicast Group Address G:

Network Layer Multicast group address. The length
of this address is network layer specific and can be
deduced from the value of AFAM. The diagram above
illustrates a four octet IPv4 address format.

4.6.5. BLIST_TYPE 5 and 6

BLIST entries of type 5 and 6 have the following
format that Figure 21 depicts.

Precedence:

8 bit unsigned integer containing the precedence
with which traffic bearing this tag will be serviced
by the TSR that issued the TDP_PIE_BIND. [Note that
the precedence is likely to be restricted to perhaps
three bits of the space reserved here.]

HC:

Hop count. See [Davie] for a detailed description.

Tag:

Tag is a 32 bit signed integer encoding the value of the tag. (See section 2.1).

Pre Len:

This one octet unsigned integer contains the length in bits of the address prefix that follows.

Prefix:

A variable length field containing an address prefix whose length, in bits, was specified in the previous (Pre Len) field. A Prefix is padded with sufficient trailing zero bits to cause the end of the field to fall on an octet boundary.

4.7. TDP_PIE_REQUEST_BIND

TDP_PIE_REQUEST_BIND is sent from a TSR to a peer to request a binding for one or more specific NLRIs, or to request all the bindings that its peer has.

A TSR receiving a TDP_PIE_REQUEST_BIND must respond with a TDP_PIE_BIND or with a TDP_PIE_NOTIFICATION. A TSR that issues a TDP_PIE_BIND in response to a TDP_PIE_REQUEST_BIND places the Request ID from TDP_PIE_REQUEST_BIND in the Request ID field in the TDP_PIE_BIND that it issues.

When a TSR receiving a TDP_PIE_REQUEST_BIND is unable to satisfy it because of resource limitations it issues a TDP_PIE_NOTIFICATION for RESOURCE_LIMIT containing the Request ID from the TDP_PIE_REQUEST_BIND.

A TSR that issues TDP_PIE_NOTIFICATION with
RESOURCE_LIMIT set must send a subsequent
TDP_PIE_NOTIFICATION, containing the status notifica-
tion RESOURCES, to the peer to whom it previously sent
that TDP_PIE_NOTIFICATION when it has resources avail-
able to satisfy further TDP_PIE_BIND_REQUESTs from
that peer.

If a TDP_PIE_NOTIFICATION is received containing
RESOURCE_LIMIT the TSR may not issue further
TDP_PIE_REQUEST_BINDs until it receives a
TDP_PIE_NOTIFICATION with the Optional parameter
RESOURCES.

A TSR may receive a TDP_PIE_REQUEST_BIND for a pre-
fix for which there is no entry in its router informa-
tion base (RIB). If this occurs the TSR issues a
TDP_PIE_NOTIFICATION containing the Optional parameter
NO_ROUTE. The value field of the NO_ROUTE parameter
contains the prefix(es) for which no entry was found
in the RIB.

The procedures to be employed by a TSR that receives
a TDP_PIE_NOTIFICATION with the optional parameter
NO_ROUTE are outside the scope of this specification.

A TSR may issue TDP_PIE_BIND and
TDP_PIE_NOTIFICATION containing RESOURCE_LIMIT or
NO_ROUTE in response to a single TDP_PIE_REQUEST_BIND.
A TSR must satisfy as much of a TDP_PIE_REQUEST_BIND
as it can. A TSR may not ignore other prefixes in a
TDP_PIE_REQUEST_BIND on encountering an error with one
prefix.

This PIE has the format that Figure 22 depicts.

TYPE:

Type field as described above. Set to 0x300 for
TDP_PIE_REQUEST_BIND.

5   LENGTH:

Length in octets of the value field of this PIE.
LENGTH is set to the length of the whole PIE in oc-
tets minus four.

10   Request ID:

This four octet unsigned integer contains a locally
significant non zero value that a TSR uses to iden-
tify TDP_PIE_BINDs or TDP_PIE_NOTIFICATIONs that are
generated in response to this request.

15

AFAM:

This 16 bit integer contains a value from ADDRESS
FAMILY NUMBERS in Assigned Numbers [Reynolds] that
encodes the address family that the network layer

20   address in the tag bindings in the BINDING_LIST is
from. This version of TDP supports IPv4 and IPv6.

ALIST_TYPE:

This 16 bit integer contains a value from the table

25   below that encodes the format of the ALIST entries
in the ADDR_LIST field. Currently there are 3 val-
ues defined by this specification.

ALIST_TYPE   ALIST entry format

30   0   Null list

1   Precedence followed by variable length
        NLRI

2   Precedence, Hop Count followed by variable
        length NLRI

The format for these entries is defined below.

ALIST_LENGTH:

Two octet unsigned integer that encodes the length
in octets of the ADDR_LIST field.

ADDR_LIST:

A variable length list consisting of one or more en-
tries of type ALIST_TYPE.

Optional Parameters:

This variable length field contains zero or more op-
tional PIEs supplied in TLV structures.

4.7.1. ALIST formats

ALIST_TYPE = 0 indicates a null list ie there are
no ALIST entries. A TDP receiving a
TDP_PIE_REQUEST_BIND with ALIST_TYPE set to 0 inter-
prets this as an implicit request for all the bindings
that it currently has.

For ALIST_TYPE = 1 ALIST entries have the form that
Figure 23 illustrates.

For ALIST_TYPE = 2 ALIST entries have the form that
Figure 24 illustrates:

HC:

Hop count.

Precedence:

This one octet unsigned integer encodes the prece-

dence with which the requestor wants traffic to this

prefix handled.

Pre Len:

This one octet unsigned integer contains the length

in bits of the address prefix that follows.

Prefix:

A variable length field containing an address prefix

whose length, in bits, was specified in the previous

(Pre Len) field. A Prefix is padded with sufficient

trailing zero bits to cause the end of the field to

fall on an octet boundary.

4.7.2. Errors

Errors are reported using TDP_PIE_NOTIFICATION.
See Figure 25.

RESOURCE_LIMIT:

If the TSR is unable to provide a TDP_PIE_BIND in

response to a request the TSR indicates this by sup-

plying the RESOURCE_LIMIT status notification as a pa-

rameter in the TDP_PIE_NOTIFICATION. The Request ID

from the the TPD_PIE_REQUEST bind is supplied in the

Value field of this status notification

RESOURCES:

A TSR that has sent RESOURCE_LIMIT to a peer sends RESOURCES when that resource limit clears.

HOP_COUNT_EQUALLED:

An ATM_TSR that receives a TDP_PIE_BIND_REQUEST containing a HOP_COUNT that equals MAX_HOP_COUNT does not generate a binding but instead sends this error notification. The length is variable and the value returns the Request ID and the ALIST entry(ies) that caused the error in the format that Figure 26 illustrates.

NO_ROUTE:

A TSR that has no RIB entry for a prefix that it receives in a TDP_PIE_REQUEST_BIND issues a notification containing this parameter for that prefix(es). The value field of this parameter contains the Request_ID, AFAM, ALIST_TYPE from the TDP_PIE_REQUEST_BIND and a suitably modified ALIST_LENGTH and ADDR_LIST in the format that Figure 27 illustrates.

See section 4.7 for descriptions of the Request_ID,AFAM, ALIST_TYPE, ALIST_LENGTH and ADDR_LIST elements.

4.8. TDP_PIE_WITHDRAW_BIND

TDP_PIE_WITHDRAW_BIND is issued by a TSR that originally provided a binding containing the tag in question and is an absolute instruction to the TSR that receives it that it may not continue to use that tag to forward traffic to the TSR issuing the TDP_PIE_WITHDRAW_BIND.

This PIE has the following format that Figure 28 illustrates.

TYPE:

Type field as described above. Set to 0x400 for TDP_PIE_WITHDRAW_BIND.

LENGTH:

Length in octets of the value field of this PIE. LENGTH is set to the length of the whole PIE in octets minus four.

BLIST_TYPE

This 16 bit integer encodes the format of the BLIST entries in the BINDING_LIST field. Possible values are defined in Section 4.6. A TDP receiving this PIE with the BLIST_TYPE set to Null interprets it (based on the semantics) as either (a) an implicit instruction to WITHDRAW all bindings belonging to the peer that issued the PIE, or (b) as an indication that all the bindings requested by the peer are no longer needed by the peer that issued the PIE.

BLIST_LENGTH:

This 16 bit unsigned integer encodes the length in octets of the BINDING_LIST.

BINDING_LIST:

Variable length field consisting of one or more BLIST entries of the type indicated by BLIST_TYPE.

The format of these entries is defined in Section 4.6.

Optional Parameters:
This variable length field contains zero or more optional PIEs supplied in TLV structures.

4.9. TDP_PIE_RELEASE_BIND

TDP_PIE_RELEASE_BIND is issued by a TSR that received a tag as a consequence of an Upstream Request/downstream assignment sequence. It is an indication to the TSR that receives it that the TSR that requested the binding no longer needs that binding.

This PIE has, with the exception of a different type value exactly the same syntax as TDP_PIE_WITHDRAW_BIND.

See Figure 29.

See the discussion of TDP_PIE_WITHDRAW_BIND for details of the syntax.

Optional Parameters:
This variable length field contains zero or more optional PIEs supplied in TLV structures.

4.10. TDP_PIE_KEEP_ALIVE

The Hold Timer mechanism described earlier in Sections 3 and 4 is reset every time a TDP_PDU is received. TDP_PIE_KEEP_ALIVE is provided to allow reset of the Hold Timer in circumstances where a TDP has no other information to communicate to its peer.

A TDP must arrange that its peer sees a TDP_PDU from it at least every HOLD_TIME period. That PDU may be any other from the protocol or, in circumstances where there is no need to send one of them, it must be TDP_PIE_KEEP_ALIVE.

This PIE has the format that Figure 30 depicts.

TYPE:

Type field as described above. Set to 0x500 for TDP_PIE_KEEP_ALIVE.

LENGTH:

Length in octets of the value field of this PIE. LENGTH is set to the length of the whole PIE in oc- tets minus four.

Optional Parameters:

This variable length field contains zero or more op- tional PIEs supplied in TLV structures.

4.11. TDP_PIE_NOTIFICATION

TDP_PIE_NOTIFICATION is issued by TDP to inform its peer of a significant event. 'Significant events' in- clude errors and changes in TSR capabilities or op- erational state.

All notification information is encoded as TLVs in the optional parameters field.

This PIE has the format that Figure 31 depicts.

TYPE:

Type field as described above. Set to 0x600 for
TDP_PIE_NOTIFICATION

LENGTH:

5   Length in octets of the value field of this PIE.
LENGTH is set to the length of the whole PIE in oc-
tets minus four.

Optional Parameters:

10   This variable length field contains zero or more op-
tional parameters supplied in TLV structures.
The optional parameter types and their uses are:

RETURNED_PDU:
15   A TSR uses this parameter to return a PDU to
the TSR that issued it.
See Figure 32.

As much as possible of the complete PDU, in-
20   cluding the header, that is to be returned is
inserted into the value field. The Length is
set to the the number of octets of the PDU that
is being returned that have been inserted into
the Value field of this optional parameter. Im-
25   plementations parsing RETURNED_PDU must be
careful to recognize that the returned PDU may
have been truncated.

CLOSING: A TSR uses this parameter to indicate
30   that it is terminating the TDP session.
See Figure 33.

TDP may send a TDP_PIE_NOTIFICATION with CLOSING set in response to a protocol error or to administrative intervention.

A TDP receiving or issuing this notification transitions to the INITIALIZED state.

The following optional parameters are defined for returning errors from individual PIEs. See the description of the relevant PIEs for a complete description of the errors.

TDP_PIE_OPEN: See Figure 34.

TDP_PIE_REQUEST_BIND: See Figure 35.

5. Intellectual Property Considerations

Cisco Systems may seek patent or other intellectual property protection for some or all of the technologies disclosed in this document. If any standards arising from this document are or become protected by one or more patents assigned to Cisco Systems, Cisco intends to disclose those patents and license them on reasonable and non-discriminatory terms.

6. Acknowledgments

Jim Gibson, Keith McCloghrie, Alex Raj, Dan Tappan and Bob Thomas pointed out omissions and errors in the previous version of this document and provided guidance on the definition of new capabilities.

7. References

[Deering] Deering, S. et al "An Architecture for Wide Area Multicast Routing", Pro Sigcomm 94 in Computer Communications Review Vol 24 No 4.

[Davie] Davie, B. et al "draft-davie-tag-switching-atm-01.txt"

[Heinanen] Heinanen, J. "Multiprotocol Encapsulation over ATM Adaptation Layer 5" RFC1483, July 1993

[Rekhter] Rekhter, Y. et al "draft-rfced-tag-switching-overview- 00.txt".

[Reynolds] Reynolds J, Postel J. "Assigned numbers" RFC 1700, October 1994

8. Author Information

Paul Doolan cisco Systems, Inc. 250 Apollo Drive. Chelmsford, MA 01824 Phone: (508) 244-8917 email: pdoolan@cisco.com

Bruce Davie cisco Systems, Inc. 250 Apollo Dr. Chelmsford, MA 01824 email: bsd@cisco.com

Dave Katz Juniper Networks, Inc. 3260 Jay St. Santa Clara, CA, 95054 email: dkatz@jnx.com

Yakov Rekhter cisco Systems, Inc. 170 Tasman Dr. San Jose, CA 95134 email: yakov@cisco.com

Eric Rosen cisco Systems, Inc. 250 Apollo Dr. Chelmsford, MA 01824 email: erosen@cisco.com

INTERNET DRAFT: MULTIPROTOCOL EXTENSIONS FOR BGP-4

Network Working Group	Tony Bates
Internet Draft	Cisco Systems
Expiration Date: March 1998	Ravi Chandra
	Cisco Systems
	Dave Katz
	Juniper Networks

Yakov Rekhter

Cisco Systems

Multiprotocol Extensions for BGP-4

draft-ietf-idr-bgp4-multiprotocol-01.txt

1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts
are working documents of the Internet Engineering Task
Force (IETF), its areas, and its working groups. Note
that other groups may also distribute working docu-
ments as Internet-Drafts.

Internet-Drafts are draft documents valid for a
maximum of six months and may be updated, replaced, or
obsoleted by other documents at any time. It is inap-
propriate to use Internet-Drafts as reference material
or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft,
please check the ``1id-abstracts.txt'' listing con-
tained in the Internet-Drafts Shadow Directories on
ftp.is.co.za (Africa), nic.nordu.net (Europe), mun-
nari.oz.au (Pacific Rim), ds.internic.net (US East
Coast), or ftp.isi.edu (US West Coast).

2. Abstract

Currently BGP-4 [BGP-4] is capable of carrying rout-
ing information only for IPv4 [IPv4]. This document
defines extensions to BGP-4 to enable it to carry
routing information for multiple Network Layer proto-
cols (e.g., IPv6, IPX, etc...). The extensions are

backward compatible - a router that supports the extensions can interoperate with a router that doesn't support the extensions.

5 3. Overview

The only three pieces of information carried by BGP-4 that are IPv4 specific are (a) the NEXT_HOP attribute (expressed as an IPv4 address), (b) AGGREGATOR (contains an IPv4 address), and (c) NLRI (expressed as 10 IPv4 address prefixes). This document assumes that any BGP speaker (including the one that supports multiprotocol capabilities defined in this document) has to have an IPv4 address (which will be used, among other things, in the AGGREGATOR attribute). Therefore, to 15 enable BGP-4 to support routing for multiple Network Layer protocols the only two things that have to be added to BGP-4 are (a) the ability to associate a particular Network Layer protocol with the next hop information, and (b) the ability to associated a par- 20 ticular Network Layer protocol with NLRI. To identify individual Network Layer protocols this document uses Address Family, as defined in [RFC1700].

One could further observe that the next hop information (the information provided by the NEXT_HOP attrib- 25 ute) is meaningful (and necessary) only in conjunction with the advertisements of reachable destinations - in conjunction with the advertisements of unreachable destinations (withdrawing routes from service) the next hop information is meaningless. This suggests 30 that the advertisement of reachable destinations should be grouped with the advertisement of the next

hop to be used for these destinations, and that the advertisement of reachable destinations should be segregated from the advertisement of unreachable destinations.

To provide backward compatibility, as well as to simplify introduction of the multiprotocol capabilities into BGP-4 this document uses two new attributes, Multiprotocol Reachable NLRI (MP_REACH_NLRI), and Multiprotocol Unreachable NLRI (MP_UNREACH_NLRI). The first one (MP_REACH_NLRI) is used to carry the set of reachable destinations together with the next hop information to be used for forwarding to these destinations. The second one (MP_UNREACH_NLRI) is used to carry the set of unreachable destinations. Both of these attributes are optional and non-transitive. This way a BGP speaker that doesn't support the multiprotocol capabilities will just ignore the information carried in these attributes, and will not pass it to other BGP speakers.

4. Multiprotocol Reachable NLRI - MP_REACH_NLRI (Type Code 14):

This is an optional non-transitive attribute that can be used for the following purposes:

(a) to advertise a feasible route to a peer

(b) to permit a router to advertise the Network Layer address of the router that should be used as the next hop to the destinations listed in the Network Layer Reachability Information field of the MP_NLRI attribute.

(c) to allow a given router to report some or all
of the Subnetwork Points of Attachment (SNPAs) that
exist within the local system

The attribute contains one or more triples <Address
Family Information, Next Hop Information, Network
Layer Reachability Information>, where each triple is
encoded as shown in Figure 36.

The use and meaning of these fields are as follows:
Address Family Identifier:
This field carries the identity of the Network
Layer protocol associated with the Network Ad-
dress that follows. Presently defined values for
this field are specified in RFC1700 (see the Ad-
dress Family Numbers section).
Subsequent Address Family Identifier:
This field provides additional information about
the type of the Network Layer Reachability In-
formation carried in the attribute.
Length of Next Hop Network Address:
A 1 octet field whose value expresses the length
of the "Network Address of Next Hop" field as
measured in octets
Network Address of Next Hop:
A variable length field that contains the Net-
work Address of the next router on the path to
the destination system
Number of SNPAs:
A 1 octet field which contains the number of
distinct SNPAs to be listed in the following

fields. The value 0 may be used to indicate
that no SNPAs are listed in this attribute.

Length of Nth SNPA:

A 1 octet field whose value expresses the length
of the "Nth SNPA of Next Hop" field as measured
in semi-octets

Nth SNPA of Next Hop:

A variable length field that contains an SNPA of
the router whose Network Address is contained in
the "Network Address of Next Hop" field. The
field length is an integral number of octets in
length, namely the rounded-up integer value of
one half the SNPA length expressed in semi-
octets; if the SNPA contains an odd number of
semi-octets, a value in this field will be pad-
ded with a trailing all-zero semi-octet.

Network Layer Reachability Information:

A variable length field that lists NLRI for the
feasible routes that are being advertised in
this attribute. When the Subsequent Address Fam-
ily Identifier field is set to one of the values
defined in this document, each NLRI is encoded
as specified in the "NLRI encoding" section of
this document.

The next hop information carried in the
MP_REACH_NLRI path attribute defines the Network Layer
address of the border router that should be used as
the next hop to the destinations listed in the MP_NLRI
attribute in the UPDATE message. When advertising a
MP_REACH_NLRI attribute to an external peer, a router
may use one of its own interface addresses in the next

hop component of the attribute, provided the external peer to which the route is being advertised shares a common subnet with the next hop address. This is known as a "first party" next hop. A BGP speaker can advertise to an external peer an interface of any internal peer router in the next hop component, provided the external peer to which the route is being advertised shares a common subnet with the next hop address. This is known as a "third party" next hop information. A BGP speaker can advertise any external peer router in the next hop component, provided that the Network Layer address of this border router was learned from an external peer, and the external peer to which the route is being advertised shares a common subnet with the next hop address. This is a second form of "third party" next hop information.

Normally the next hop information is chosen such that the shortest available path will be taken. A BGP speaker must be able to support disabling advertisement of third party next hop information to handle imperfectly bridged media or for reasons of policy.

A BGP speaker must never advertise an address of a peer to that peer as a next hop, for a route that the speaker is originating. A BGP speaker must never install a route with itself as the next hop.

When a BGP speaker advertises the route to an internal peer, the advertising speaker should not modify the next hop information associated with the route. When a BGP speaker receives the route via an internal link, it may forward packets to the next hop address

if the address contained in the attribute is on a common subnet with the local and remote BGP speakers.

An UPDATE message that carries the MP_REACH_NLRI must also carry the ORIGIN and the AS_PATH attributes (both in EBGP and in IBGP exchanges). Moreover, in IBGP exchanges such a message must also carry the LOCAL_PREF attribute. If such a message is received from an external peer, the local system shall check whether the leftmost AS in the AS_PATH attribute is equal to the autonomous system number of the peer than sent the message. If that is not the case, the local system shall send the NOTIFICATION message with Error Code UPDATE Message Error, and the Error Subcode set to Malformed AS_PATH.

5. Multiprotocol Unreachable NLRI - MP_UNREACH_NLRI (Type Code 15):

This is an optional non-transitive attribute that can be used for the purpose of withdrawing multiple unfeasible routes from service.

The attribute contains one or more triples <Address Family Information, Unfeasible Routes Length, Withdrawn Routes>, where each triple is encoded as shown below:

```
+-----------------------------------------------+
| Address Family Identifier (2 octets           |
+-----------------------------------------------+
| Subsequent Addres`s Family Identifier (1 octet)|
+-----------------------------------------------+
| Withdrawn Routes (variable)                   |
+-----------------------------------------------+
```

The use and the meaning of these fields are as fol-
lows:

Address Family Identifier:

This field carries the identity of the Network
Layer protocol associated with the NLRI that
follows. Presently defined values for this field
are specified in RFC1700 (see the Address Family
Numbers section).

Subsequent Address Family Identifier:

This field provides additional information about
the type of the Network Layer Reachability In-
formation carried in the attribute.

Withdrawn Routes:

A variable length field that lists NLRI for the
routes that are being withdrawn from service.
When the Subsequent Address Family Identifier
field is set to one of the values defined in
this document, each NLRI is encoded as specified
in the "NLRI encoding" section of this document.

An UPDATE message that contains the MP_UNREACH_NLRI
is not required to carry any other path attributes.

6. NLRI encoding

The Network Layer Reachability information is en-
coded as one or more 2-tuples of the form <length,
prefix>, whose fields are described below:

```
+---------------------------+
|    Length (1 octet)       |
+---------------------------+
|    Prefix (variable)      |
+---------------------------+
```

The use and the meaning of these fields are as fol-
lows:

a) Length:

The Length field indicates the length in bits
of the address prefix. A length of zero indi-
cates a prefix that matches all (as specified by
the address family) addresses (with prefix, it-
self, of zero octets).

b) Prefix:

The Prefix field contains address prefixes fol-
lowed by enough trailing bits to make the end of
the field fall on an octet boundary. Note that
the value of trailing bits is irrelevant.

7. Subsequent Address Family Identifier

This document defines the following values for the
Subsequent Address Family Identifier field carried in
the MP_REACH_NLRI and MP_UNREACH_NLRI attributes:

1 - Network Layer Reachability Information used for
unicast forwarding

2 - Network Layer Reachability Information used for
multicast forwarding

3 - Network Layer Reachability Information used for
both unicast and multicast forwarding

This document reserves values 128-255 for vendor-
specific applications.

This document reserves value 0.

8. Security Considerations

Security issues are not discussed in this document.

9. Acknowledgements

To be supplied.

10. References

[BGP-4]

[IPv4]

[IPv6]

[RFC1700]

11. Author Information

Tony Bates Cisco Systems, Inc. 170 West Tasman Drive
San Jose, CA 95134 email: tbates@cisco.com

Ravi Chandra Cisco Systems, Inc. 170 West Tasman
Drive San Jose, CA 95134 email: rchandra@cisco.com

Dave Katz Juniper Networks, Inc.

3260 Jay St. Santa Clara, CA 95054 email:
dkatz@jnx.com

Yakov Rekhter Cisco Systems, Inc. 170 West Tasman
Drive San Jose, CA 95134 email: yakov@cisco.com

INTERNET DRAFT: CAPABILITIES NEGOTIATION

Network Working Group   Ravi Chandra

Internet Draft   Cisco Systems

Expiration Date: February 1998   John G. Scudder

                                         Internet Engineering Group, LLC

Capabilities Negotiation with BGP-4

draft-ietf-idr-bgp4-cap-neg-00.txt

1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

2. Abstract

Currently BGP-4 [BGP-4] requires that when a BGP speaker receives an OPEN message with one or more unrecognized Optional Parameters, the speaker must terminate BGP peering. This complicates introduction of new capabilities in BGP.

This document defines new Optional Parameter, called Capabilities, that is expected to facilitate introduction of new capabilities in BGP by providing graceful capability negotiation.

The proposed parameter is backward compatible - a router that supports the parameter can maintain BGP

peering with a router that doesn't support the parameter.

3. Overview of Operations

When a BGP speaker that supports capabilities negotiation sends an OPEN message to its BGP peer, the message includes an Optional Parameter, called Capabilities. The parameter lists the capabilities supported by the speaker. The speaker can mark a listed capability as "Required", which means that if the peer doesn't recognize/support the capability, the BGP peering shall be terminated.

When the peer receives the OPEN message, if the message contains the Capabilities Optional Parameter, the peer checks whether it supports all of the listed capabilities marked as R, and if not, sends a NOTIFICATION message, and terminates peering. The Error Subcode in the message is set to Unsupported Capability. The message should contain all the capabilities marked as R that are not supported by the peer. If the peer doesn't support a capability that is not marked as R, the peer should not use this as a reason to terminate peering.

A BGP speaker may use a particular capability when peering with another speaker if both speakers support that capability. A BGP speaker determines the capabilities supported by its peer by examining the list of capabilities present in the Capabilities Optional Parameter carried by the OPEN message that the peer sends to the speaker.

A BGP speaker determines that its peer doesn't support capabilities negotiation, if in response to an OPEN message that carries the Capabilities Optional Parameter, the speaker receives a NOTIFICATION message with the Error Subcode set to Unsupported Optional Parameter.

4. Capabilities Optional Parameter (Parameter Type 2):

This is an Optional Parameter that is used by a BGP speaker to convey to its BGP peer the list of capabilities supported by the speaker.

The parameter contains one or more triples <Capability Code, Capability Length, Capability Value>, where each triple is encoded as shown below:

```
+------------------------------+
| Capability Code (1 octet)    |
+------------------------------+
| Capability Length (1 octet)  |
+------------------------------+
| Capability Value (variable)  |
+------------------------------+
```

The use and meaning of these fields are as follows:

Capability Code:

Capability Code is a one octet field that unambiguously identifies individual capabilities.

The high-order bit of this field is used to mark the capability as "Required" (if the bit is set to 1).

Capability Length:

Capability Length is a one octet field that contains the length of the Capability Value field in octets.

Capability Value:

Capability Value is a variable length field that is interpreted according to the value of the Capability Code field.

5. Extensions to Error Handling

This document defines new Error Subcode - Unsupported Capability. The value of this Subcode is 7. The Data field in the NOTIFICATION message lists the set of capabilities that are marked as Required, but are either unsupported or unrecognized by the BGP speaker that sends the message. Each such capability is encoded the same way as it was encoded in the received OPEN message.

6. Security Considerations

Security issues are not discussed in this document.

7. Acknowledgements

To be supplied.

8. References

[BGP-4]

9. Author Information

Ravi Chandra Cisco Systems, Inc. 170 West Tasman Drive San Jose, CA 95134 e-mail: rchandra@cisco.com
John G. Scudder Internet Engineering Group, LLC 122 S. Main, Suite 280 Ann Arbor, MI 48104 e-mail: jgs@ieng.com

What is claimed is: